# A high-throughput system for automated training combined with continuous long-term neural recordings in rodents

# A high-throughput system for automated training combined with continuous long-term neural recordings in rodents

A dissertation presented

by

**Rajesh Poddar**

to

**The Division of Medical Sciences**

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

In the subject of

Neurobiology

Harvard University

Cambridge, Massachusetts

April 2015

Dissertation Advisor: Dr. Bence P. Ölveczky Rajesh Poddar

# A high-throughput system for automated training combined with continuous long-term neural recordings in rodents

## Abstract

Addressing the neural mechanisms underlying complex learned behaviors requires training animals in well-controlled tasks and concurrently measuring neural activity in their brains, an often time-consuming and labor-intensive process that can severely limit the feasibility of such studies. To overcome this constraint, we developed a fully computer-controlled general purpose system for high-throughput training of rodents. By standardizing and automating the implementation of predefined training protocols within the animal's home-cage our system dramatically reduces the efforts involved in animal training while also removing human errors and biases from the process. We deployed this system to train rats in a variety of sensorimotor tasks, achieving learning rates comparable to existing, but more laborious, methods. By incrementally and systematically increasing the difficulty of the task over weeks of training, rats were able to master motor tasks that, in complexity and structure, resemble ones used in primate studies of motor sequence learning. We also developed a low-cost system that can be attached to the home-cages for recording neural activity continuously in an unsupervised fashion for the

entire months-long training process. Our system allows long-term tethering of animals and is designed for recording and processing tens of terabytes of raw data at very high speeds. We developed a novel spike-sorting algorithm that allows us to track the activity of many simultaneously recorded single neurons for weeks despite large gradual changes in their spike waveforms. This is done with minimal human input enabling, for the first time, the identification of almost every single spike from a single neuron over many weeks of training. We used these systems to record from the motor cortex of rats as they learned to perform a sequence of highly stereotyped movements. We found that neural activity in the motor cortex was exquisitely correlated with the behavior. Surprisingly, the pattern of neural activity in the motor cortex was similar before and after learning despite the fact that motor cortex is required to learn the task, but not to perform it once it has been acquired.

## Table of Contents

## Table of Figures

## Acknowledgements

I would like to thank John Assad, David Cox, Markus Meister, and Rachel Wilson – my dissertation advisory committee members – for guiding me over the years in completing my dissertation and continually encouraging me to do the best that I can. I am also grateful to the Office of Animal Resources, Eny Moldanado in particular, for enthusiastically working with us despite our unconventional requirements. I am thankful for all the engineering support provided by hackers extraordinaire, Ed Soucy and Joel Greenwood. The seven years that I took to complete my dissertation was made infinitely more enjoyable by all the conversations and discussions I had with members of the Ölveczky, Cox, and Meister labs, and other affiliates of the Center for Brain Science. A special thank you goes to all friends for sitting through day after day of lunch conversations with me. I am also grateful to my family for continuing to support me through the ups and downs of graduate school.

I reserve the deepest gratitude for my advisor, Bence Ölveczky for having confidence in me and allowing me to craft a very independent graduate school experience. I am also very grateful to him for providing me a family and for inviting me into his home. His support through difficult decisions and his broad perspective was critical to finishing this dissertation. I feel extremely lucky to have found him for an advisor, mentor, colleague and friend – I couldn't have been more fortunate.

## Introduction

The human brain is an impressively complex network of ~10^14 connections between ~10^11 neurons that underlies our capacity for reasoning, ingenuity and other forms of complex behavior[1]. By filtering and processing the massive amounts of often ambiguous and unstructured incoming sensory information in sophisticated ways, it guides our actions and enables continued learning. No artificial system comes close to achieving the level of performance of ordinary humans even in such mundane tasks as invariant object recognition[2] or dexterous manipulation of objects[3]. Furthermore, human brains achieve this performance while consuming several orders of magnitude less power than artificial systems attempting to replicate it[4]. As such, understanding the principles by which the organization and function of neural circuits lead to perception and behavior has been of interest not only to scientists wishing to understand normal brain function but also to engineers interested in replicating it in silico.

The brain structures of non-human primates in particular and mammals in general bear remarkable anatomical similarity to humans. As such one of the dominant experimental paradigms in neuroscience for more than half a century has been to record the activity of neurons in the brains of non-human primates as they learn and perform complex perceptual, cognitive and motor tasks [5–7]. A typical multi-year research project under this paradigm led by a graduate student or a postdoc often involves training a handful of animals (typically rhesus monkeys) in moderately complex tasks and recording from dozens to hundreds of neurons in hour-long recording sessions. These experiments are very time consuming and labor intensive - training animals often takes months, recording typically require close supervision by the researcher – yet produce relatively small amounts of data especially compared to the enormous complexity of the object under investigation.

Lowering the barriers to studying brain function and behavior in non-human animals by automating the labor intensive parts of these experiments, and even of data pre-processing steps like spike sorting, in a low-cost manner promises to increase the throughput of these experiments and the

scale of data collection by at least an order of magnitude, thus accelerating progress toward making neuroscientific discoveries. We contend that many subfields of neuroscience, including behavioral neurophysiology, are at present data-starved – the amount and nature of data that individual experiments collect is not large enough to substantially constrain the universe of hypotheses about how networks of neurons might coordinate to compute perceptions, decisions and behavior. Furthermore, the difficulty of these experiments often makes replication challenging increasing the time-scale for weeding out spurious results. Finding how general a particular finding is by repeating the experiment under different conditions and in different contexts would also be greatly aided by developing systems to make neuroscience experiments high-throughput.

While many previously labor intensive experiments in molecular biology and genomics have now been automated and commoditized[5–7], neuroscience has only just started to see meaningful progress in this direction[8]. This includes the development of microfluidic devices for whole-organism imaging for worms[9], flies[10], and fish[11], automated sectioning and imaging of neural tissue at nanometer resolution in order to reconstruct the connectivity structure of whole brains[12], and, robots for automated patch-clamping[8]. Many of these techniques generate an unprecedented amount of data and have the potential to revolutionize our understanding of brain function once good data analysis tools are developed.

In this thesis, we describe our contribution to this tradition of developing fully or largely automated high-throughput systems to radically reduce the barriers for doing certain kinds of experiments. In Chapter 1, we describe a low-cost system for fully automated training of rodents in a high-throughput manner in their home-cages that requires no manual intervention beyond standard animal care. In Chapter 2, we describe a system for continuous months-long neural recordings in behaving animals and a largely automated algorithm for tracking the activity of populations of single neurons for

the duration of the recordings. Finally in Chapter 3, we use the systems described in the previous two chapters to study the neural representation of learned motor sequences in the motor cortex of rats.

## A fully automated high-throughput training system for rodents

### Introduction

Studies exploring the neural mechanisms underlying higher-order cognitive and learning phenomena, including decision making[13], motor skill execution[14], and perceptual discrimination[15,16], have traditionally been done in non-human primates. Costs and regulations, however, make high-throughput experiments on monkeys difficult to justify[17]. Rodents, with their increasingly well-understood cognitive and learning capabilities, have emerged as an alternative model system for studying a variety of complex behaviors[18–27]. Rats and mice share the basic mammalian brain architecture with primates, and though cortical specializations may differ[28–30], recent studies suggest that many of the well-characterized cortical functions in primates have equivalents in rodents[19,21,24,25,27,31]. Sophisticated tools for measuring and manipulating brain activity[32–35] together with the many transgenic lines and disease models available in rodents further incentivize their use in mechanistic studies. Yet one of the main barriers for research on complex behaviors, both in rodents and primates, lies in training animals - a process typically done under close supervision of researchers who frequently modify protocols and procedures on an animal-by-animal basis to improve learning rates and performance. This approach is labor-intensive and time-consuming, and makes the interaction between animal and researcher an integral part of the training process, possibly confounding comparisons of experimental outcomes across animals and labs[36].

Here we describe a method and experimental infrastructure that fundamentally transforms this traditionally arduous process, making it effortless, rigorous and amenable to high-throughput approaches. Our solution combines two main ingredients.

1. *Automation of the training process*. Automation allows implementation of rigorously defined training protocols on a large scale without the vagaries and efforts associated with human-

assisted training[37]. Such improvements in the quality and quantity of behavioral data enables powerful research approaches for addressing complex and slow-to-learn behaviors [31].

2. *Training within the animal's home-cage.* Implementing automated training within the animal's home-cage eliminates the need for day-to-day handling of trained animals, making long training processes fully automated and largely effortless. Live-in home-cage training also enables long-term tethering of animals, making long-term uninterrupted neural recordings feasible.

While certain aspects of animal training have been automated, either through the use of custom-developed software or commercially available systems[31,38–42], most solutions lack the flexibility required to tackle broader sets of questions or behaviors. Existing solutions do not accommodate either complete automation of multi-stage training processes involving large numbers of animals or long-term neural recordings in the context of training. Thus the significant human involvement currently required for experiments on complex behaviors still represents a considerable impediment to large-scale rodent studies.

To further improve the efficiency of the training process, we developed a fully Automated Rodent Training System (ARTS) for reward-based learning (Figure 1.1). Our low-cost system is flexible, extensible, remotely administrable, and allows for simultaneous training of large cohorts of animals. ARTS is designed for deployment in animal facilities found in biomedical research centers, and requires no more human supervision than standard animal care. Training occurs in custom-designed home-cages that can be outfitted with a variety of sensors, manipulanda, water ports, and effectors (e.g. sensory stimulation devices), customized to the nature of the behavioral task (Figure 1.1A; see Supplementary Video S1 for a demonstration).

Below, we outline the general architecture and logic of ARTS and its current implementation and describe, validate, and benchmark its use for motor learning in rats.

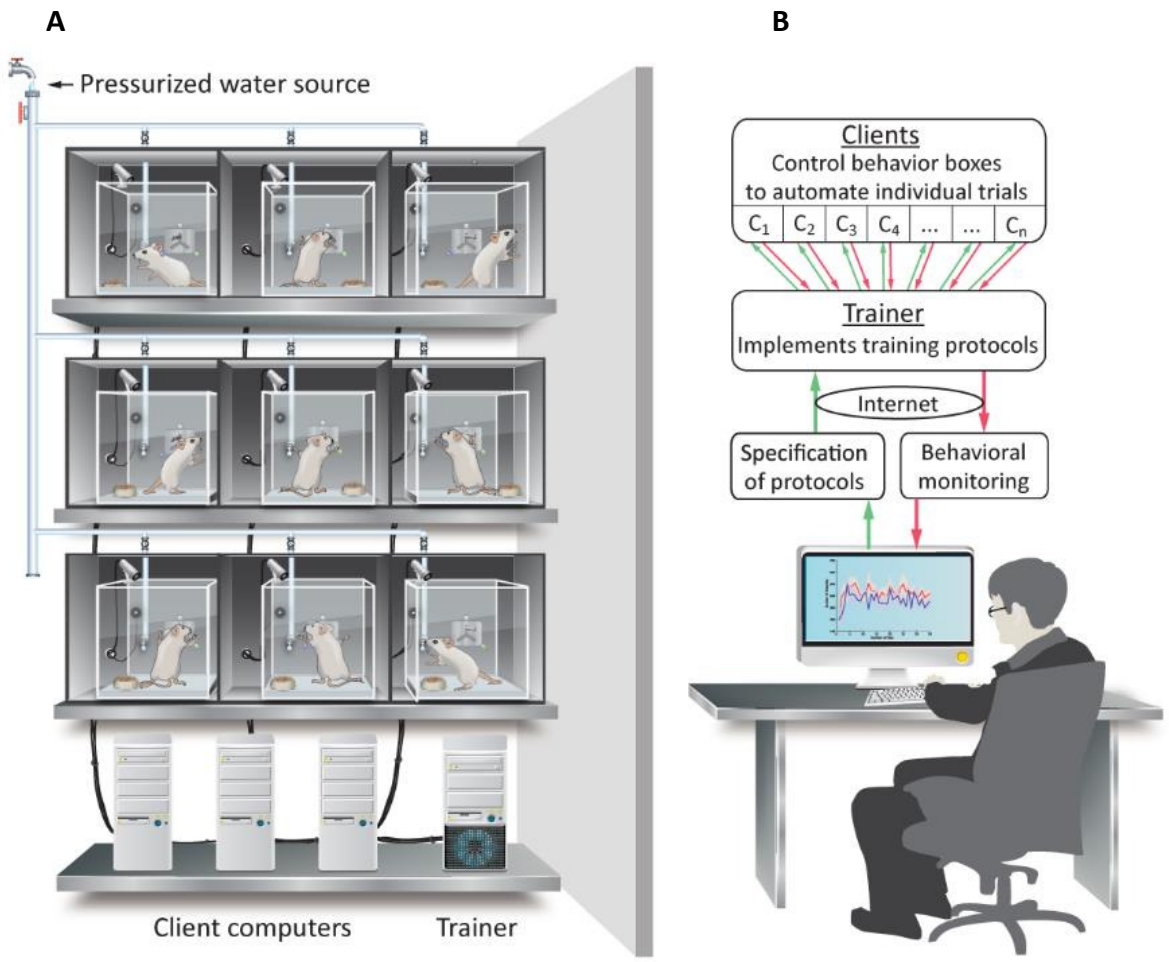



**Figure 1.1:** A fully automated high-throughput training system for rodents. **A.** Schematic of the hardware implementation. Custom-made behavior boxes, housing individual rats, are outfitted with task-specific sensors, manipulanda, and effectors. Water reward is delivered through a computer-controlled solenoid valve connected to the animal facility's pressurized water supply. *Client* computers (2 boxes per computer) directly control and monitor the behavior boxes via a data acquisition card, using rules supplied by the *Trainer*. **B**. Logic of the software implementation. Training protocols and behavioral monitoring is implemented through a flexible and hierarchical software architecture. *Clients* (c) control individual trials by directly interfacing to a behavior box. The *Trainer* monitors overall performance and implements the training protocol by informing *Clients* about the rules and structure of individual trials. The user controls and monitors training remotely over the internet via a graphical user interface. Green arrows denote information flow regarding the training protocol; red arrows represent flow of behavior data.

## System Architecture

*Flexible and modular software architecture for control of high-throughput animal training*

The heart of our automated rodent training system is the software platform that interacts with individual home-cages and executes pre-specified training protocols (Figure 1.1B, Figures S1.1 – S1.3 in File S1). To allow for maximum flexibility and generality, the software suite is modular and hierarchical, with two different components controlling distinct aspects of the training process (Figure 1.1B). At the top of the hierarchy is the *Trainer,* which monitors overall performance and implements user-defined training protocols for individual behavioral boxes. A 'protocol' is defined as a set of training stages and performance criteria for automatically transitioning between them (see Figures S1.5, S1.6, and Methods in File S1 for details). Each training stage is specified in the form of a finite state machine (FSM), a widely employed and intuitive abstraction for specifying behavioral tasks that consists of states linked by transitions[43]. Behavioral or environmental events, such as lever presses, nose pokes, or elapsed time, can trigger transitions between states, each of which can be associated with a set of actions (e.g. reward being dispensed, LEDs turning on/off).

The *Trainer* executes training protocols by supplying FSMs specifying reward contingencies and trial structure to lower level *Clients*, each of which controls a behavioral box. With this flexible and general program structure, automating a training protocol in ARTS reduces to having the *Trainer* supply the *Client* with the right FSMs at the right times.

Data acquired by the *Client* during the execution of an FSM, including high-resolution timing data and video, is stored in a central database. This allows multiple users to concurrently and efficiently read and write data to and from the database using SQL - an intuitive database language supported by all major programming languages. Centralized data storage also allows for easy backup, aggregation, analysis, and distribution of large amounts of behavioral data. A software package for ARTS complete with a user's manual can be downloaded from our server.

*Software Implementation*

The entire software suite for ARTS is written in C#.Net, a simple, general-purpose, object-oriented programming language. The software (both source code and pre-compiled binaries) along with detailed step-by-step instructions on setting up the system can be downloaded from http://olveczkylab.fas.harvard.edu/ARTS (the system is co-branded OpCon in the website and internally). Both the *Client* and *Trainer* can easily be extended to accommodate virtually any behavioral task or training protocol by simply writing add-on custom-scripts in any .NET compatible language, including C#, F#, J#, VB, and C++. Writing or using these plugins does not require a detailed understanding of the underlying software (the plugins and scripts included in the source code can serve as a starting point). Importantly, the software supports numerous extensions to basic FSMs, like custom plugins and concurrent execution of multiple FSMs[43], which enables specification of behavioral paradigms with probabilistic cues and complex reward contingencies.

In addition to the core components of ARTS (*Client*, *Trainer*, and the database), a suite of supporting software adds further functionality, making it a complete end-to-end high-throughput automated training system (Figures S1.1-S1.3 in File S1). A graphical user interface allows for easy and intuitive specification of training protocols. Behavioral monitoring, including querying the timing, duration, and saved video of each behavioral 'event' is made possible by an interactive data visualization tool. Furthermore, a suite of network services & scripts enables remote control and monitoring (including live streaming video onto the internet) of the system.

*A cost-effective hardware solution*

The hardware requirements of ARTS are modest, making it cost-effective and easy to build and deploy (Figure 1.1, Figure S1.1 in File S1). Our current system contains 48 behavior boxes controlled by 24 *Client* computers and two servers, all of which are housed in a temperature and humidity controlled animal facility. The reward port providing water reinforcement in each behavior box is connected to the animal facility's pressurized water system through solenoid valves, allowing us to dispense specific

volumes of water by controlling the duration of valve openings through the *Client*. Aquarium pumps (Jehmco LPH 60) are used to ventilate behavior boxes at a ratio of one pump for every six boxes. To ensure acoustic and visual separation each box is placed in an enclosure (Supplementary Figure 1.4A in File S1), which is placed on the shelves of a standard wire racks.

Behavior boxes were custom designed using acrylic and aluminum extrusions. The boxes have a removable front panel holding all experimental equipment (sensors, indicators, manipulanda, water dispensing valves etc.; Figure S1.4A in File S1). The front panel is the only part of the system that needs to be customized for a given experimental paradigm. To ensure compatibility with invasive experiments (chronic electrophysiology etc.), the normal lid of the box can be exchanged with a custom lid having the experiment-specific equipment (e.g. commutator etc.). The hardware cost for building the boxes is ~$500/box.

Each low-cost *Client* computer runs Windows 7 and contains an Intel quad-core processor (Core i5-750 - 2.66GHz/core), 4GB of DDR3 SDRAM, and a 1.5TB 5900 RPM hard disk and controls and communicates with the behavior boxes at the ratio of two boxes per computer. Behavioral data (e.g. from manipulanda, lick sensors, and cameras) and signals for controlling peripherals such as speakers and LEDs are transferred between the box and the *Client* computer via a National Instruments data acquisition system (NI PCIe 6323 - DAQ card, 2 x RC68-68 – Ribbon Cable & 2 x CB-68LP – Connector Block). The *Client* computers, in turn, communicate with a central server, which runs the *Trainer* and hosts the database (Figure 1.1). Server computers (*Trainer*) contain a higher-end Intel quad-core processor (Core i7-950 – 3.06GHz/core), 12GB of DDR3 SDRAM, 3 1TB 7200 RPM hard disks. The servers run Windows Server 2008 R2 and host a SQL Server 2008 R2 database.

The total cost of our current ARTS set-up (48 boxes), including all hardware, computers, and electronics, is around $67K, i.e. $1400/box (see Supplementary Table 1 in File S1 for a breakdown of the costs and the “Hardware” section of http://olveczkylab.fas.harvard.edu/ARTS for a detailed list of the

components of ARTS). Supplementary Video S1 shows ARTS deployed in our animal facility. Detailed designs and specifications of the hardware implementation are available upon request.

*Scalability and Safety*

The number of behavior boxes per *Client* computer is primarily limited by the number of video cameras attached to each box since acquisition/processing of video data is fairly CPU intensive. ARTS supports multiple data acquisition cards per *Client* computer allowing a large number of behavior boxes to be controlled via a single computer if the *Client* does not need to process video data. Likewise the network bandwidth is dominated by uncompressed video data (17MBps per 30fps 640x480 webcam). The server can be scaled to support a larger number of clients by increasing the amount of available memory since this is the bottleneck for database performance.

In addition to free water at the end of every night (see section *Schedules and mode of reinforcement*) multiple layers of security checks are built into the system to prevent animals from dehydrating. Water is only dispensed upon licking the reward spout ensuring that the dispensed water is consumed. The behavior monitoring GUI prominently displays water consumption. A watchdog program continually monitors the *Client*, *Trainer* and database to detect any failures and displays this information in the monitoring GUI. Finally animals are periodically examined and weighed by the animal care staff to ensure they are adequately hydrated.

## Behavioral Training Methods

We have used *ARTS* to train more than 150 rats in a variety of behaviors including pressing a lever in precise temporal sequences, pressing a set of levers in spatiotemporal sequences on cue, and moving a joystick in various directions on cue. In this report we focused on a subset of 67 female Long Evans rats aged ~10-12 weeks at the start of the experiment, 30 of which were trained to perform a simplified version of the center-out task (Figures 1.2, 1.3) and 39 on a precise lever pressing task (Figure 1.4). Animals were kept on a daily 12h light cycle.

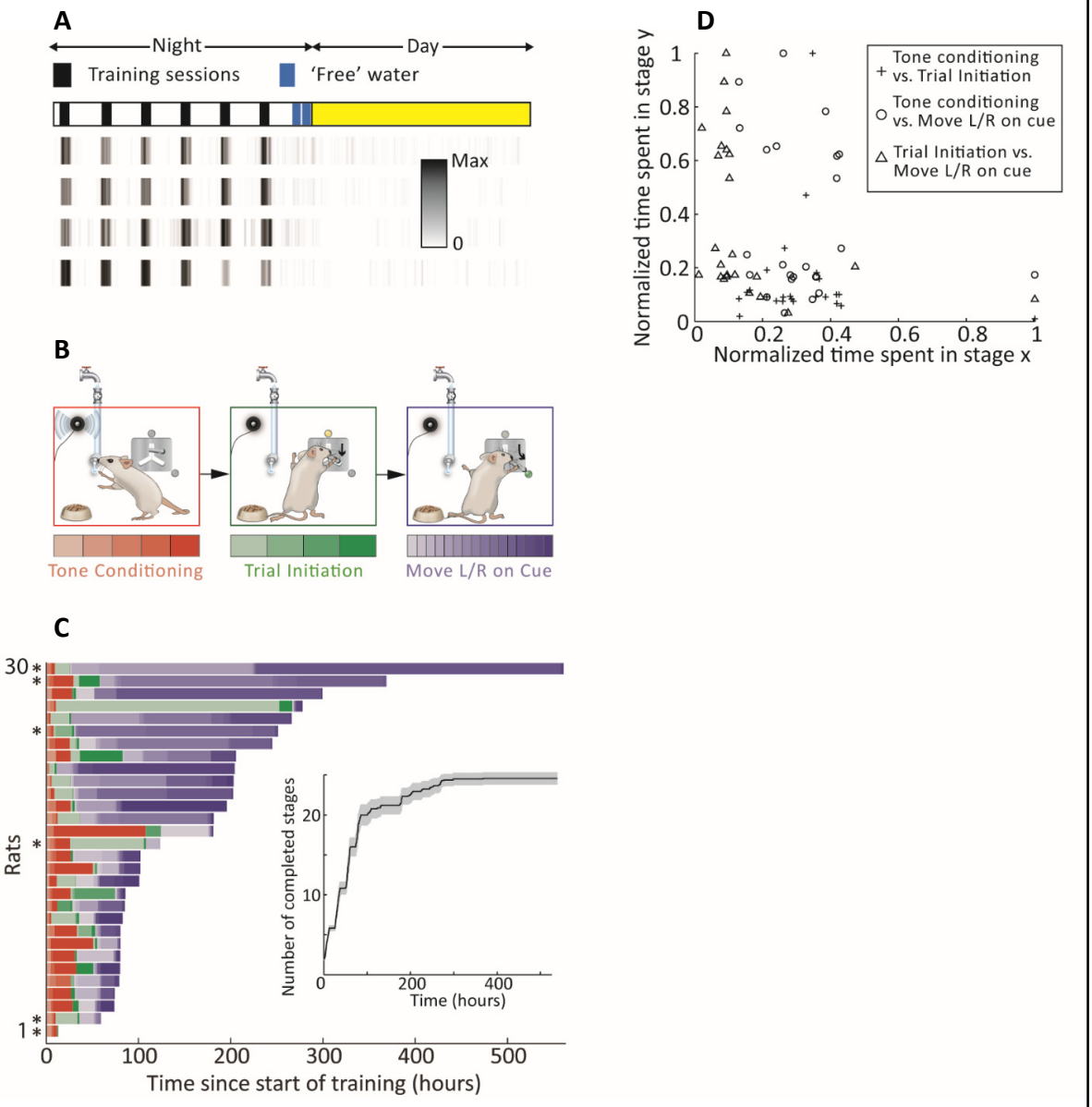



**Figure 1.2:** Complete automation of a complex multi-step training protocol for a version of the center-out movement task (see Methods). **A**. Training session structure. Animals were trained during six nightly 30-minute training sessions. The density plots show the distribution of joystick presses for four representative animals in their third week of training. 'Free' water is only available to rats earning less than a minimal amount of water during the nightly training session. **B**. Thirty rats were trained to perform the center-out movement task in three successive stages, each with multiple sub-stages (see Supplementary Methods in File S1 for details). Stage 1: touching the joystick for a reward tone and subsequently licking at the water spout to initiate water reward delivery. Stage 2: moving the joystick down on cue. Stage 3: moving the joystick left and right. **C**. Stage and sub-stage completion times for each rat. Six rats (indicated with asterisks) were dropped from the study due to poor learning. Inset shows the mean (and standard error) of the number of completed sub-stages as a function of time. **D**. Time needed to complete one stage vs. another for the 24 successful rats.

*Schedules and mode of reinforcement*

Naïve rats were water-deprived for 8-10 hours before being transferred to their behavior boxes. After this, rats were trained for the next several weeks automatically by *ARTS* with no human involvement in the day-to-day training process. For the tasks in Figures 1.2 and 1.3, rats had 30-minute training sessions during their subjective night every 2 hours for a total of six training sessions. For the task in Figure 1.4, rats had 2 60-minute sessions per night. At the end of the night, ARTS automatically dispensed free water up to their daily minimum (5ml per 100g body weight). Rats also had a rest day every week during which water was dispensed *ad libitum*. Blinking house lights, a continuous 10s 1kHz pure tone and a few drops of water marked the beginning of each training and free water session.

*Center-out task*

Rats were trained to move a 2D joystick left/right along two arms of an inverted-Y shaped slit (Figure 1.2). The equilibrium position of the joystick is at the top of the inverted-Y. Trial availability was indicated by the center LED; a rat could commence a trial by moving the joystick down by ~2 cm to the point where the two arms of the inverted-Y meet. Then, the left (right) LED turned on, and if the rat guided the joystick >5 cm along the left (right) arm of the slit, the trial was considered successful and a reward tone (1000Hz for 100ms) presented. The rat could then lick the reward spout to collect water and commence the next trial. If the rat moved in the wrong direction, a 7 second timeout was instituted before the next trial could be initiated.

*Precise lever pressing task*

Rats were trained to press a lever twice with a 700 ms delay between the presses. Animals could self-initiate the trial by pressing the lever. After learning to associate lever pressing with water, rats were rewarded for increasingly precise approximations to the target sequence, i.e. 2 lever presses separated by 700ms. The reward contingencies were automatically updated based on performance to ensure that, on average, ~30-40% of the trials were rewarded. If the trial was successful, a reward tone was played

and water reward dispensed upon licking of the reward spout. Animals had to wait 1.2s before initiating the next trial if the inter-press interval fell outside the rewarded range.

## System Validation

ARTS is designed for fully automated training of complex behaviors in a home-cage environment and thus represents a significant departure from current practice. The day-to-day interaction of the researcher with experimental animals and training apparatus is eliminated as is the need for transferring animals back-and-forth between procedural chambers and holding rooms. Whether completely replacing the researcher with contextual cues can successfully get animals to learn complex tasks has not been evaluated. To ensure the feasibility of our approach as a general purpose solution for large-scale rodent training, we have done extensive testing and characterization of the system, including training more than hundred and fifty animals in a variety of sensorimotor tasks, a subset of which we report on below.

*Structure of fully automated training*

A major challenge presented by home-cage training is motivating animals to perform the behavioral task. In traditional reward-based training paradigms[22,27,39], the experimenter places a water- or food-deprived animal in the behavioral apparatus and rewards correct behaviors with liquids or foods. As hunger or thirst is satisfied, the researcher removes the animal and commences deprivation anew. We automate this process within the animal's home-cage by dispensing water as reward only during training sessions, the start and end of which are indicated to the animal by a set of salient sensory cues (e.g. flashing house lights). Time between sessions serves to deprive the animal of water and thus build up motivation for the next session. We have deployed a variety of session structures. For the center-out movement task described below (Figure 1.2B), for example, animals had 6 30-minute training sessions per day spaced at 2 hour intervals (Figure 1.2A), whereas for the task in Figure 1.4, we employed 3 daily 60-minute sessions, each separated by 4 hours. Whether a particular session structure is superior to

others has yet to be rigorously tested, but our experience thus far suggests that this is not a critical parameter.

Animals quickly learn to engage with the task (e.g. manipulating a joystick) predominantly during specified training sessions: in the third week of training in the center-out task, the likelihood of a rat pressing the joystick was, on average, 24 times higher in-session than out-of-session (n = 24 rats; Figure 1.2A). To prevent poorly performing animals from dehydrating, water is provided at the end of the night for animals that do not receive the prescribed minimum daily water amount during training.

*Validation of ARTS: Center-out movement task*

A standard paradigm for studying neural control of movement in primates is the center-out reaching task[44,45], which involves moving a manipulandum to one of several possible cued locations. Rats trained with traditional methods can master a version of this task in a matter of weeks[27]. In our implementation of the task, rats are required to move a two-dimensional joystick along the arms of an inverted Y-shaped slot with their forepaws (Figure 1.2B, Methods). Trials are initiated by moving the joystick down the vertical arm of the slot in response to an LED cue. A second LED then prompts the animal to move the joystick either left or right. A correct trial is indicated by a short tone followed by water reward. Thirty rats were trained on the task in three sequential stages, each containing multiple sub-stages (Figure S1.5 and Methods in File S1). All but one rat completed the first training stage (touching the joystick for a reward tone and collecting water reward) within 12 hours. Twenty-four out of 30 rats completed the second (moving the joystick vertically down on cue) and third (moving the joystick left and right) training stage to criterion (Figure 1.2C, Methods). Despite being trained using the same training protocol, animals learned at different rates (time to complete all three stages = 148 ± 78 hrs (mean ± S.D.) from start of training, range = 73 – 299 hrs, n = 24 rats). Furthermore, learning rates on one training stage was not a good predictor for mastery of other stages, which involved different sets of cognitive, learning

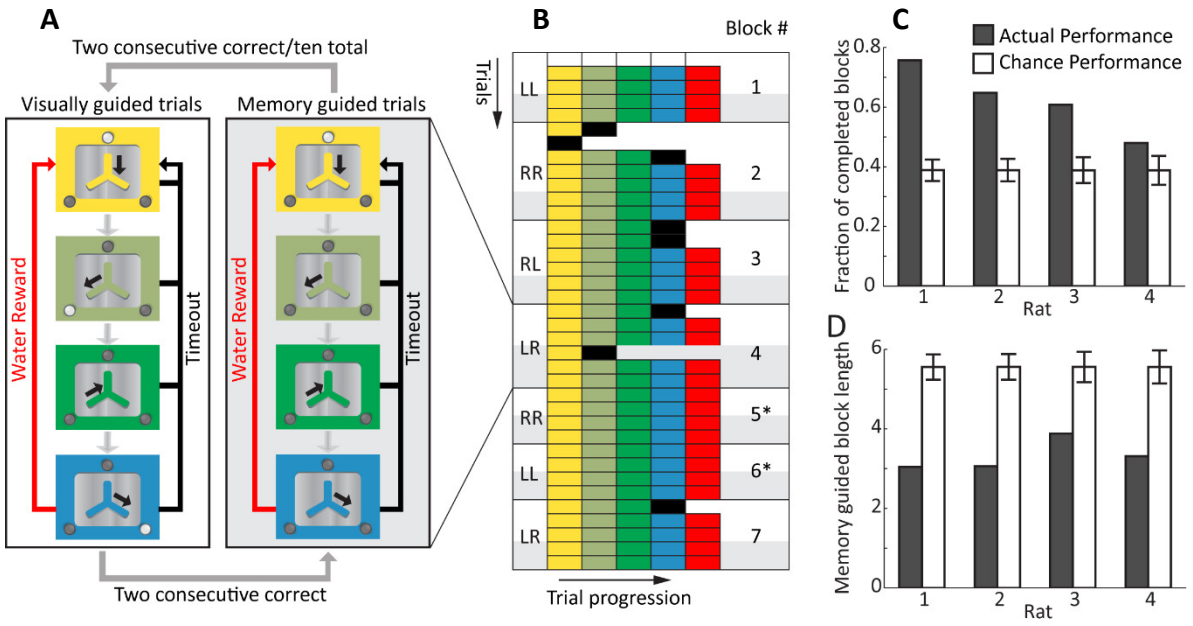



**Figure 1.3:** Automated training of memory guided action sequences. **A**. Structure of the behavioral task. An experimental block starts with a visually guided trial (left), in which the center LED indicates trial initiation. Upon moving the joystick down, the left (right) LED comes on. After a successful left (right) movement the LED turns off and the joystick is moved back to the center. A second movement is then cued by the right (left) LED. Upon moving the joystick right (left) a water reward is delivered. Any erroneous movement results in a timeout. After two consecutive correct trials, directional cues are not given and the movement sequence has to be performed from memory (right). After two consecutive correct memory guided trials (or ten total trials – an *incomplete* block), the next block commences with a new sequence. **B**. A sample run of 7 consecutive blocks from one animal. Each row represents one trial, with the sequence of movements color coded as in 'A'. Left column denotes the target sequence (L-left movement; R-right movement). Shaded trials denote memory guided trials. Blocks denoted with asterisk correspond to perfect performance. Supplementary Video S2 shows experimental blocks 5-7. **C-D.** Aggregate performance of 4 rats trained in the task as measured by the fraction of completed blocks (**C**) and the number of memory guided trials until completion (**D**). Performance is compared to simulated chance (error-bars denote 95% confidence level). Data from 679 blocks for Rat 1, 647 blocks for Rat 2, 472 blocks for Rat 3 and 392 blocks for Rat 4.

and motor control challenges. The correlation coefficients between the time to complete different stages were -0.05 (stage 1, 2), -0.18 (stage 2, 3), and -0.34 (stage 1, 3) respectively (n = 24 rats; Figure 1.2D). Faced with such a substantial variation in the speed of learning across subjects and in distinct phases of learning, studies on complex learning that use learning rate as a behavioral readout will require large cohorts of animals trained in identical tasks, an approach that will be much helped by automated high-throughput training systems.

*Validation of ARTS: Memory guided motor sequence execution*

Having the capacity to simultaneously and effortlessly train large groups of animals, reduces the risk associated with - and the investment made in – individual animals, making it feasible to train even very challenging tasks, i.e. ones that only a small fraction of animals may be capable of learning. We deployed our automated training set-up to explore whether rats can master sophisticated motor sequence learning paradigms previously used only in primates[14]. In particular we were interested in the extent to which rodents can execute action sequences from working memory[46]. We trained the 4 best rats in the center-out task (Figure 1.2) to make sequences of left/right joystick movements from memory (Figure 1.3A, Figure S1.6 in File S1). At the beginning of each block of trials, visual cues (LEDs) were used to instruct the correct sequence of movements. After 2 consecutive correct visually guided trials, cues were removed and animals had to perform the same movement sequence from memory. Rats progressed to the next block (i.e. new sequence) after 2 consecutive correct memory guided trials or 10 trials, whichever occurred first.

Figure 1.3B shows an example of the star performer in this task once asymptotic performance was reached (see also Supplementary Video S2). The errors in the visually guided trials at the start of some blocks are typically due to the animal performing the sequence from the prior block.

To measure the extent of learning, we compared a week's performance on the task to simulated chance (Figures 1.3C, 1.3D), modeled as random left/right movements during the memory guided trials. All 4 rats completed significantly more blocks (i.e. got 2 consecutive correct memory guided trials within a span of 10 trials) than expected by chance (Figure 1.3C, fraction of completed blocks = 63% vs. 39% by chance; probability of observing performance by chance < 2e-4). Furthermore, the average number of memory guided trials required to complete a block (which can range from 2 to 10) was substantially smaller than chance levels for each animal (Figure 1.3D, 3.2 vs. 5.4 for chance; probability of observing performance by chance < 1e-4). The best rat completed over 75% of the blocks with, on average, only 3

memory guided trials per block. These results validate rodents as a model for working memory guided motor sequence generation, and ARTS as an efficient method for training such complex behaviors.

*Benchmarking home-cage training against existing training methods*

To benchmark the live-in training concept against more traditional methods, we compared the performances of rats trained in our home-cage set-up (n = 24 rats) with ones housed in social groups and exposed to the behavior apparatus only during daily training sessions (n = 13 rats) (Figure 1.4A). Both groups were trained in identical behavioral boxes using the same automated training protocol. Rats were trained to spontaneously press a lever twice with a 700 ms delay between presses (Figure 1.4A, Methods). Motivation to do the task, as measured by the number of trials initiated per day, was similar between the two groups (494 ± 243 (mean ± S.D.) trials per day for automated training vs. 426 ± 123 trials per day for manual training on day 15 of training; Figure 1.4B). Furthermore, learning rates, as characterized by the fraction of 'correct' trials (defined here as inter-press intervals within 30% of the 700 ms target) at 20,000 trials was also comparable (83% ± 8% for automated training vs. 76% ± 17% for manual training, p=0.22; Figure 1.4C). Beyond demonstrating the feasibility and non-inferiority of live-in training in terms of performance, our results also validate the use of our automated training system in cases when rats are transferred to behavior boxes only for the duration of training. While home-cage training has the obvious advantage of requiring no human involvement other than standard animal care, there may be scenarios in which the benefits of fully automated home-cage training outweigh the negative effects of social isolation[47,48]. In such instances ARTS can still automate all other aspects of training (as was done for the socially housed cohort in the precise lever pressing task, Figure 1.4). An added benefit of manually transferring animals to the behavior box during training sessions is that the same box can be multiplexed across many animals increasing the throughput of the system[31,38].

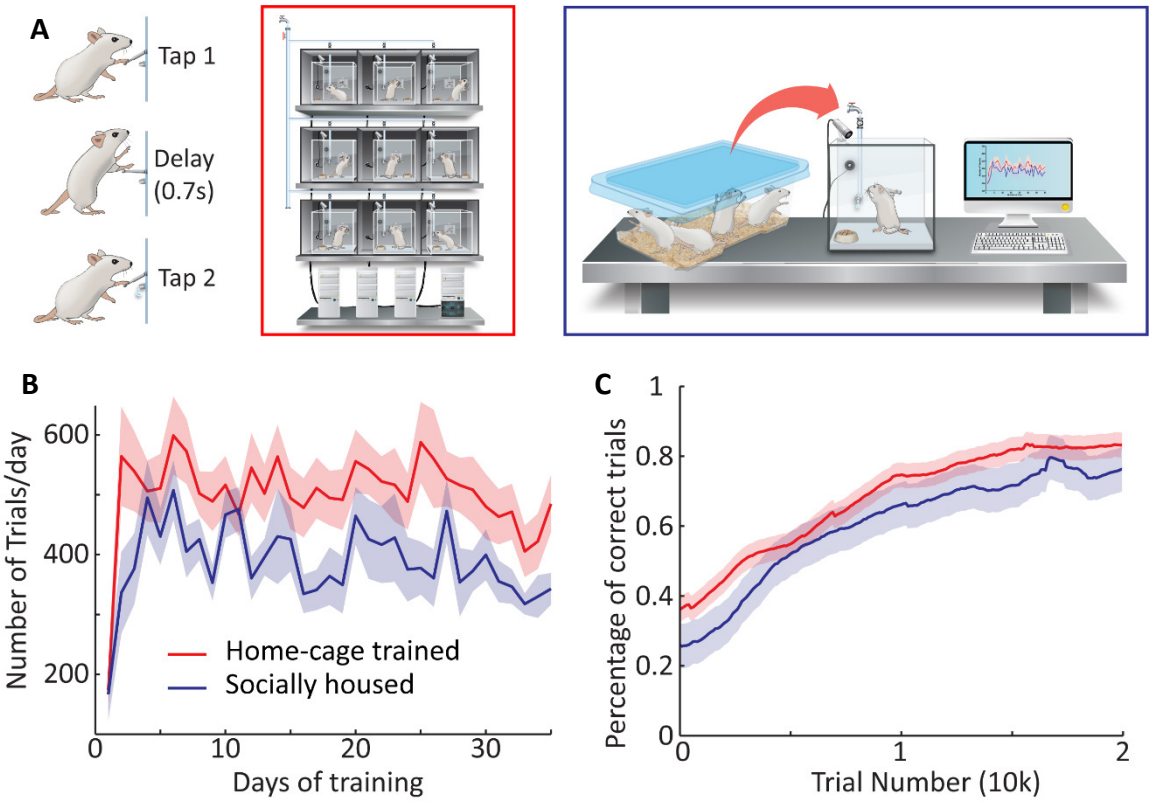



**Figure 1.4:** Fully automated live-in home-cage training is comparable to existing methods in terms of learning and performance. **A**. Rats were trained to spontaneously press a lever twice with a 700 ms delay between presses in an individually housed live-in training paradigm (red, n = 24 rats) or in a socially housed setting in which they were transferred to the behavior apparatus for daily training sessions (blue, n = 13 rats). **B**. Motivation as measured by the number of trials per day over time. **C**. Learning performance as measured by the fraction of correct trials, defined as trials within 30% of the 700 ms target inter-press interval, over time. Shaded regions in B and C represent standard error across animals.

## Discussion

We present a cost-effective, modular, and fully automated training system for rodents (ARTS, Figure 1.1) that dramatically decreases the effort required for implementing operant learning paradigms. Deploying the system in our animal facility enabled high-throughput training of rats with performance and learning rates similar to more traditional methods (Figure 1.4). While we benchmarked our system in a variety of motor learning tasks (Figures 1.2, 1.3 and 1.4), we believe that its flexibility, modularity, and extensibility ensures that it can be used to automate virtually any training protocol relying on reward-based learning. Though we designed and benchmarked ARTS for rats, a simple modification to the

geometry of the home-cage should make the system applicable also to mice, though the extent to which mice are amenable to fully automated training in our system remains to be seen.

Simple behavioral tests in rodents have revolutionized our understanding of neurological function by allowing large-scale phenotyping of experimental animals[49]. Automated training further extends the power of rodent models in neuroscience by enabling standardized high-throughput studies of more complex behaviors[42]. Full automation also removes the vagaries inherent to human-assisted training by requiring explicit codification of all training steps, including contingencies and criteria for progressing from one stage to the next (example in Figures S1.5, S1.6 and Methods in File S1). Such a compact and complete description of the training process makes reproducing and comparing experimental outcomes across different animal cohorts and labs possible and meaningful.

Automated training protocols not only standardize the training process, but they ensure that incremental insights and improvements to training strategy accumulate. Indeed, our experience in setting up novel training tasks is that their implementation improves with time, as inefficiencies and 'bugs' in the training protocol get sorted out. In contrast to human-assisted training, where these experiential gains are largely confined to the researcher, automated training ensures that each improvement becomes part of an ever-evolving protocol.

Having well defined discrete training stages, each associated with its distinct set of cognitive, learning, perceptual, or motor control challenges, also enables increased specificity of the behavioral analysis. For example, when we analyzed learning rates in different phases of a multi-stage task we found no correlation between them, meaning that facility with associative learning aspects of a task, for example, may not translate into success on motor learning aspects (Figure 1.2D). Breaking down the learning process to its elementary components by evaluating each training stage independently will permit a more detailed phenotypic analysis and thus help better pinpoint how specific manipulations, genetic or otherwise, impact learning and performance of complex multi-faceted behaviors.

The advantages of home-cage training go beyond the benefits of full automation. It eliminates animal handling and the performance variability that goes with it[37] and fully automated continuous long-term neural recordings in behaving animals a feasible prospect. An initial practical concern with home-cage training, however, was the possible impact of social isolation on learning and performance[47,48]. In our benchmarking, however, we did not see a difference in either motivation or learning rates as compared to animals that were housed socially and exposed to the behavioral chamber only during training (Figure 1.4). It is possible that any detrimental effect of social isolation is compensated for by other factors unique to automated training, such as precise and regimented training schedules. Further experiments are needed to fully characterize the effects of social isolation on motivation and learning in a home-cage setting, with the understanding that different tasks may be impacted differently.

Lowering the barrier for training large number of animals on complex behavioral tasks, as ARTS does, has the potential to accelerate research towards understanding many fundamental questions in neuroscience.

## Supporting Information Legends

**Supplementary Video S1**: A 2m30sec video highlighting the functionality and features of ARTS, and showing its deployment in our animal facility.

**Supplementary Video S2**: Video of a rat performing the task shown in Figure 1.4. The video contains the stretch of trials corresponding to Blocks 5-7 in Figure 1.4B. On the right of the movie file is seen the joystick trajectory. Colored cues shown in the video correspond to cues seen by the animal (obscured in the video). Red square corresponds to the cue for initiating a trial. Green square denotes the cue for pushing joystick to the right; blue square for pushing the joystick to the left.

**Supplementary File S1**: Supplementary Methods, Supplementary Table 1, and Figures S1.1 – S1.6.

## Attribution

This chapter has been published in a peer-reviewed journal[50]. Rajesh Poddar conceived of, designed, and wrote software for the system. Rajesh Poddar performed the experiments shown in Figures 1.2 and 1.3 while Risa Kawai performed the experiments shown in Figure 1.4. Rajesh Poddar did all the data analyses. Bence P Ölveczky and Rajesh Poddar together wrote the manuscript.

# A system for continuous long-term neural recordings in rodents

## Introduction

One of the primary experimental tools for understanding the relationship between neural activity and behavior is the extracellular recording of action potentials from single neurons. While this approach is often used to investigate the neural correlates of behavior at timescales of seconds, minutes, and occasionally hours, experimental techniques for tracking the activity of populations of single neurons over days and weeks remain elusive[51]. In this paper, we describe and characterize the first end-to-end fully automated system for continuous 24x7 recordings of action potentials from single units and for tracking these units stably, often for weeks, in behaving rodents.

*Advantages of continuous long-term recordings*

Recording from the same population of neurons continuously for long periods of time offers several advantages over cross-sectional comparisons between distinct groups of neurons at distinct time points. Recording from the same set of neurons at two time points or across two experimental conditions increases statistical power by making the experimental design 'within-group' rather than 'between-groups'. Furthermore, longer recordings mean that activity of single neurons can be tracked for a larger number of trials, thus allowing the detection of even very small effects. More importantly, continuous long-term recordings allow us to address a range of questions not possible with a cross-sectional approach. For example, many studies find that large fractions of neurons respond in task-specific ways to virtually any task the animal is trained on[52], raising the question whether two neurons in a brain area that are functionally similar during a given task are also functionally similar in a different task. Is there an underlying commonality to the features encoded by a neuron across tasks? How do these neurons behave in non-task related contexts like unstructured exploration or sleep? How does the pattern of neural activity and the underlying neural circuit that generate them change as a behavior is gradually learnt over

the course of many weeks? Once a behavior is learnt, does its neural representation remain stable? The ability to record from targeted brain areas stably and in a continuous manner over days and weeks would significantly help in addressing these and other important questions.

Fully automated continuous recordings also eliminates the laborious manual steps typically involved in recording from behaving animals, like transferring animals to their recording chamber, plugging and unplugging recording cables. This allows scaling recording experiments to a large number of animals and enables a single researcher to easily supervise tens or even hundreds of recordings simultaneously. Combined with the fully automated training system described in the previous chapter, this enables experiments to study the neural correlates of weeks-long learning processes with little human involvement in a high-throughput manner.

*Previous attempts at recording from the same set of units over time*

Previous studies have attempted to record stably from the same set of units for weeks by recording from chronically implanted electrode arrays during hour-long recording sessions every day. Similarity of action potential (AP) waveforms and stability of functional characteristics are then used to match units across days[53–56]. This approach is likely to have a large number of false negatives since AP waveforms change over time due to slight motion of the electrodes between recording sessions[57] and since functional properties of neurons also change with time and learning. Also, nearby neurons of the same type can be mistakenly considered to be the same leading to false positives[54]. Action potentials with similar extracellular waveforms recorded several days apart can be attributed to the same neuron with much greater confidence if recorded continuously with no gaps in between. Furthermore, action potentials with different waveforms can still be assigned to the same single unit if it is shown to continuously morph from one to the other over time.

Recently, two-photon imaging of fluorescence-based calcium indicators in mouse brains has been used to record from the same population of neurons at regular intervals for several weeks[58,59]. This has provided one of the first longitudinal dataset of neural activity. However, the temporal resolution of these indicators is several orders of magnitude worse than electrical recordings (100ms vs 1ms) and they cannot be used to record from deep brain areas without damaging overlying neural tissue[60]. Also, continuous recordings allow changes in the neural activity to be observed as they happen, enabling questions relating to how circuits re-organize with learning and the role of sleep in the plasticity of neural circuits. To date, the longest continuous recording of a single unit lasted approximately 48 hours and was conducted using a wireless system from primate brains[61].

## System Design

### LONG TERM TETHERING OF BEHAVING ANIMALS

*Rats tend to destroy the cable tether given continuous access to it*

The dominant approach to recording single units in freely behaving rodents during intermittent hour-long recording sessions does not scale to a continuous 24x7 setting for a number of reasons. Chief among these is that animals find ingenious ways of destroying the bundle of wires (primarily by chewing it up) that relay electrically measured neural activity from the brain to the data acquisition system. We verified this by leaving freely behaving rats (n=2) in custom built but standard electrophysiological recording chambers. Each animal was put in a behavior box (similar to the ones used in the automated training experiments in the previous chapter) with a multi-conductor cable connected to a commutator. The commutator allows the animal to walk around freely without twisting the cable and thus prevents the accumulation of rotational forces on the rat's head. Both animals chewed up the cable within 24 hours. Furthermore, once an animal did this, the replacement cable was destroyed within hours.

*Previous approaches and failed solutions*

One straightforward solution to this problem is to eliminate the cable from the recording setup entirely by wirelessly transmitting recorded neural data[62–66]. However, even with very low-power amplification circuitry, the amount of energy required would necessitate frequently replacing the battery pack needed to power the headstage making this solution quite labor intensive. For instance, a 64 channel wireless headstage designed for rats by Szuts et al[62] consumes 645mW of power. This would exhaust a 25g (10 percent of the weight of a rat) 1100mAh Li ion battery in about 6 hours. For mice, which weigh an order of magnitude less than rats, the battery pack would need to be replaced prohibitively frequently. An attractive solution is to power the headstage wirelessly using RF induction[66]. However, no such solution is currently commercially available. Another major drawback of wireless headstages is that power requirements scale at least linearly with the channel count making this problem even more acute for higher channel count recordings.

Another approach that researchers in the field have tried is to coat the cable with a repellant like capsaicin. However, in a setting where the animal has continuous 24x7 access to the cable, they are reported to eventually overcome their distaste for capsaicin and destroy the cable anyway. Yet another approach reported in the literature involves connecting the cable to a system of pulleys and counterweights to prevent the animal from physically reaching the cable[67]. We replicated a version of this system adapted to rodents but noticed a tendency for the strings used in the setup to get entangled with the cable as the animal moved around. Another solution that we considered, but ultimately discarded, was to make the cable into a spring with the goal of preventing the animal from being able to grasp the cable. However, for reasonable cable lengths and cage heights this would result in large forces on the animal's head since the retraction force of a spring scales linearly with its extension. However, a constant force spring, one whose retraction force is independent of the extension (like a retractable tape measure), ought to overcome this problem.

*Pulley and linear-slide based solution for long-term tethering*

Our final design emulates a constant force spring with a low-cost counterweighted linear-slide and pulley system that effectively prevents the animal from reaching the cable by always keeping the tether taut and thus out of its reach (Figure 2.1A). In our design, the recording device on the rat's head is connected via a cable to a passive commutator attached to a carriage that rides on a low-friction linear-slide (friction force < 10g). The carriage is counterweighted via a pulley, resulting in a constant but small upwards force (approx. 10g) on the tethering cable that removes all slack while still allowing the animal to freely move around its cage. The friction in the linear slide and pulley needs to be countered by the

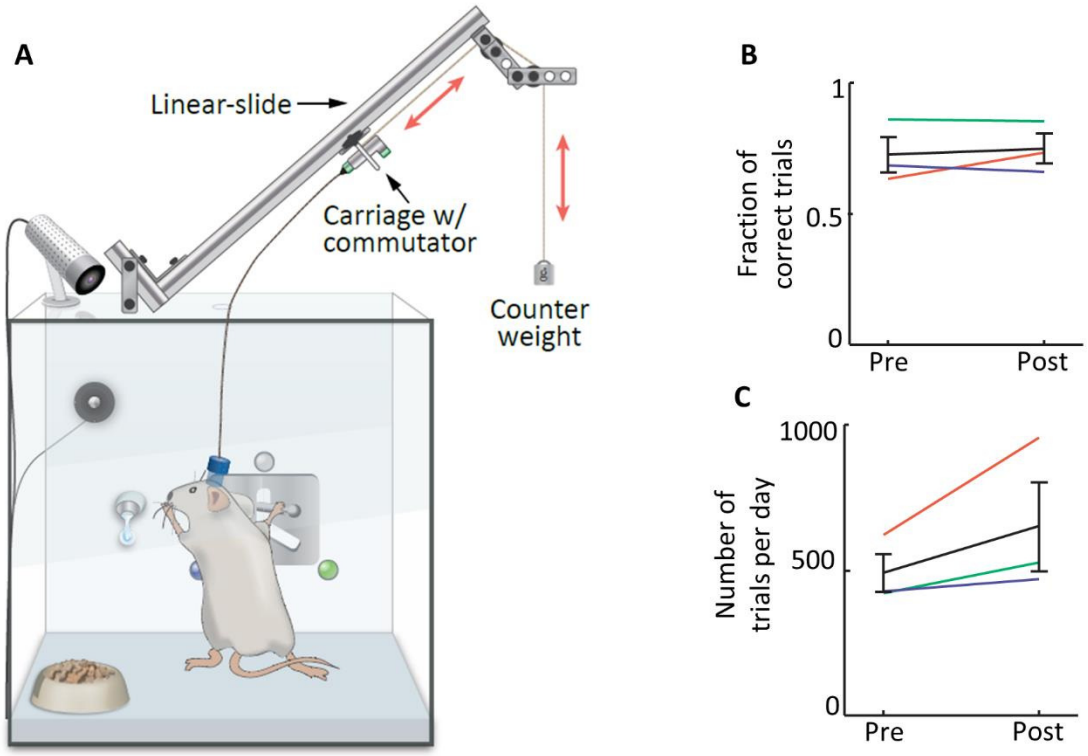



**Figure 2.1:** Home cage adapted for continuous long-term neural recordings. **A**. Schematic of the set-up. The implanted recording drive is connected, via a cable, to a passive commutator attached to a carriage that rides on a low-friction linear slide. The carriage is counterweighted by a pulley removing all slack while still allowing the animal to freely move around the cage. **B and C**. Performance (as measured by the fraction of correct trials – panel B) and motivation (as measured by the number of trials – panel C) before and after implantation and addition of pulley-system to the home cage for 3 rats in the center-out task described in Chapter 1.

animal as it moves around the behavior box, and should be minimized. In our experience a force of ~10 g does not interfere with the animal's performance (Figures 2.1B and 2.1C).

The angle at which the linear slide is mounted above the cage parametrizes a trade-off between the height of the apparatus and the maximum bend angle of the cable which in turn impacts the performance of the commutator. A perfectly vertical linear slide, while increasing the total height of the apparatus, results in only small cable bend angles and hence smooth operation of the commutator. On the other hand, a perfectly horizontal linear slide introduces large cable bend angles when the animal is in certain parts of the behavior box resulting in larger rotational forces on the animal's head before the commutator can relieve it. We have found that an approx. 45 degree angle is an acceptable compromise that works well in practice.

The functionality of a home-cage can thus be extended by replacing its top panel with a custom-made variant outfitted with the experimental infrastructure (commutator, pulley and linear-slide system) described above (Figure 2.1A). This allows animals deemed suitable for recording to be implanted with recording drives and placed back into their familiar training environment (i.e. home cage), but now with the recording extension added. These procedures (implant, adding pulley-system to the home-cage) did not adversely affect the trained behavior in any of the three rats that were tested (fraction of correct trials in the center-out task (see Chapter 1): 0.78 ± 0.13 (mean ± S.D.; pre), 0.80 ± 0.10 (post), number of total trials per day: 489 ± 114 (pre), 650 ± 267 (post), n = 3 rats; Figure 2.1B). One animal (blue line in Figure 2.1B) was continuously tethered for multiple weeks, all the while performing the task at similar levels of success and motivation as before tethering (fraction of correct trials: 0.73 (pre), 0.71 (post), number of trials per day: 423 (pre), 468 (post); Figure 2.1C).

**SIGNAL PROCESSING AND RECORDING HARDWARE**

*RHD2132 based 64-channel signal processing chain*

We also developed a novel low-cost system for high channel-count extracellular neural recordings from behaving animals by taking advantage of a recently released IC (integrated circuit) by Intan Technologies (http://www.intantech.com/). Commercially available systems for this application consist of multiple devices and a rack of electronics costing tens of thousands of dollars. Our system, contains two primary components, a headstage that remains on the animals head for the duration of the recordings and a digital interface board that is placed outside the home-cage (Figure 2.2). The headstage is a 28mm X 18mm 4-layer circuit board containing two 32-channel versions of this IC (RHD2132, Intan Technologies) that weighs less than 5 grams total. The headstage incorporates almost all aspects of the signal processing chain including digitally programmable band-pass filtering, 200-fold amplification, and analog-to-digital conversion at a rate of 30,000 samples per second per channel with 16 bits of resolution per sample. The digitized signals from each 32-channel IC is multiplexed into one high speed (84 MHz) digital signal. The output of our headstage contains a total of 12 wires and is then sent to a computer through a USB 2. 0 port via a digital interface board containing an FPGA. The digital interface board was designed to accept additional digital inputs from, for example levers, poke sensors and camera triggers to be recorded in sync with the neural data from the headstage. Likewise, the digital interface board also provides a down-sampled version of the clock used to time the sampling of the neural data that can be sent to the behavior control system. The headstage design also incorporates a 3-axis accelerometer allowing continuous monitoring of the kinematics of the animal's head enabling episodes of sleep, grooming, exploration, etc. to be detected[68]. Our system is similar to the more widely used Open-Ephys system, both of which rely on the same RHD2132 IC. Our system was designed to be aggressively low-cost and hence differs from the Open-Ephys system in the choice of connectors and cables and unlike the Open-Ephys system contains a very basic no-frills digital interface board.

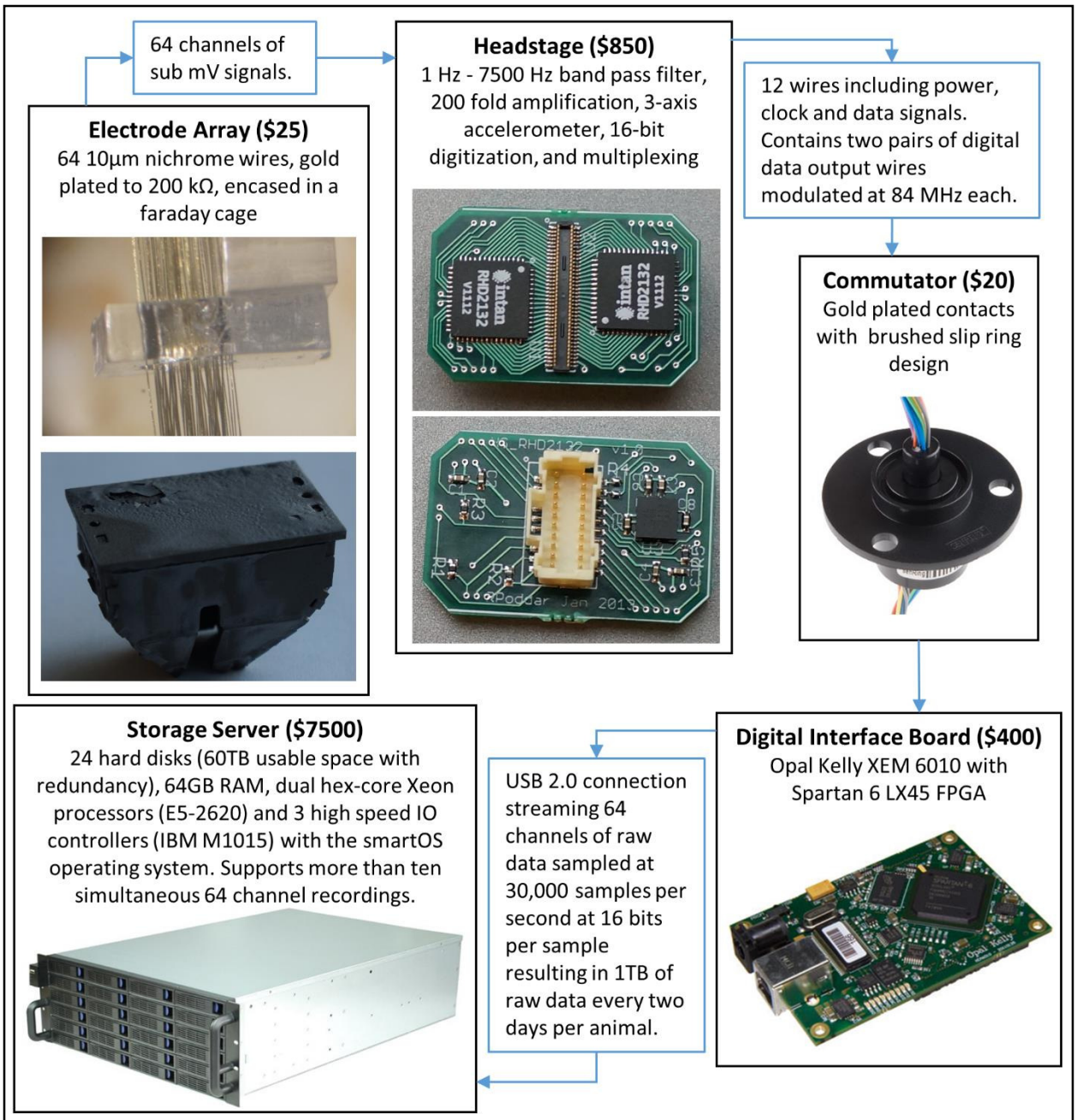



**Figure 2.2:** Scalable low-cost signal processing and recording hardware. Neural activity is recorded using an electrode array. A headstage continuously filters, amplifies, and digitizes the measured signals and sends them over to a digital interface board via a commutator. The digital interface board packages the data into USB packets and streams it to a storage server.

In addition to its low cost, our system has highly favorable noise characteristics since the electrical signals measured at the tip of the electrodes are amplified and digitized very close to their source on the animal's head. We encase the headstage in a small faraday cage (made by spray coating the protective

enclosure with silver paint) that gets implanted on the head along with the electrode array, virtually eliminating 60Hz power line noise. Furthermore, since only digital signals exit the headstage, the signal path to the computer containing long cables and a commutator are very robust to electro-magnetic interference. Because of this we were able to use cheap mass-manufactured commutators (SparkFun Electronics – ROB-13065, $20) instead of ones custom-made for scientific instrumentation that cost thousands of dollars each.

*Automated closed-loop electroplating of electrode arrays*

By taking advantage of the impedance measurement and electroplating capabilities of the RHD2132 IC we developed the first fully automated closed-loop electroplating system for multi-electrode arrays. Extracellular neural recordings in rodents typically employ an array of very thin (approx. 25µm in diameter) nichrome wires with tips that are gold plated to reduce their impedances for improved signal-to-noise ratio. This is done by bathing the electrode array in gold cyanide solution and manually alternating between impedance measurement and current injection resulting in a gradual decrease of the impedance[69]. When done manually, the amount of current injected in each pulse is very imprecise and can result in shorts between adjacent electrodes because the electrode tips accumulate too much gold. For an array of 64 electrodes this process often takes more than an hour. We designed our headstage and the firmware of our digital interface board to allow for rapid alternation between the impedance measurement and electroplating modes of the headstage. A simple software controller is then used to automatically alternate between the two modes until the target impedance is reached.

**DATA STORAGE AND COMPUTING INFRASTRUCTURE**

*Hardware setup*

One of the biggest challenges associated with continuous 24x7 recordings is storing and efficiently processing the massive amounts of raw data produced by this method. A 64 channel recording at 30,000

samples per second per channel translates to 1 terabytes (TB) of raw data every 2 days. A typical study with a few animals recorded for 2 months each results in 100s of terabytes of data. We developed a custom low-cost and, reliable, high IO bandwidth storage solution with a custom lightweight fileserver for this application (Figure 2.2). Each storage server consists of 24 4TB spinning SATA hard disks connected in parallel to a dual socket Intel server class motherboard via the high bandwidth PCI-E interface. The ZFS file-system (bundled with the open source SmartOS operating system) is used to manage the data in a redundant configuration that allows any two disks in the 24 disk array to simultaneously fail without data loss. In some situations, up to 6 disks can fail without impacting data integrity. With such a large storage array random bit-flips occasionally occur that can lead to data corruption. ZFS prevents this by periodically checking the entire dataset for such silent bit-rots. Due to the redundancy, each server has 60TB of usable space that can be read at approximately 16 gigabits per second (Gbps). This high IO bandwidth is critical for data backup, recovery and integrity verification.

*Distributed computing software setup*

The key to fully utilizing available CPU and IO resources is to process the data in parallel[70]. Thread-level parallelization inside a single process is the simplest approach and coordination between threads is orchestrated using memory shared between the threads. However, this approach only works for a single machine and does not scale to a cluster of computers. The typical approach to cluster-level parallelization is to use a map-reduce framework like Hadoop[71] that coordinates the multiple parallel computations running both within a machine and across machines in a cluster by exchanging messages between the concurrently running processes. The map-reduce abstraction conceptualizes a computation as having two phases: a ‘map’ phase which first processes small subsets of the data in parallel and a ‘reduce’ phase which then serially aggregates the results of the ‘map’ phase (Figure 2.3A). However, Hadoop was primarily designed for CPU limited workloads rather than IO limited workloads and hence requires frequent reading and writing of data to disks and across network links[72].

Since many different kinds of simple exploratory data analysis is IO limited (like computing statistics of spike waveform amplitudes), we developed a custom distributed computing infrastructure for map-reduce like computations for the .NET platform (Figure 2.3B). The major novelty in our framework is that rather than moving the output of the map computation over the network to a central node for performing the reduce computations, it instead moves the reduce computation to the machines containing the output of the map computations in the correct serial order. If the output of the map computation is voluminous compared to the output of each step of the reduce computation then our approach consumes significantly less time and IO bandwidth. We have used this framework both in a virtual cluster of 10 virtual machines running on the afore-mentioned SmartOS-based storage servers and in Microsoft's commercial cloud computing platform, Windows Azure.

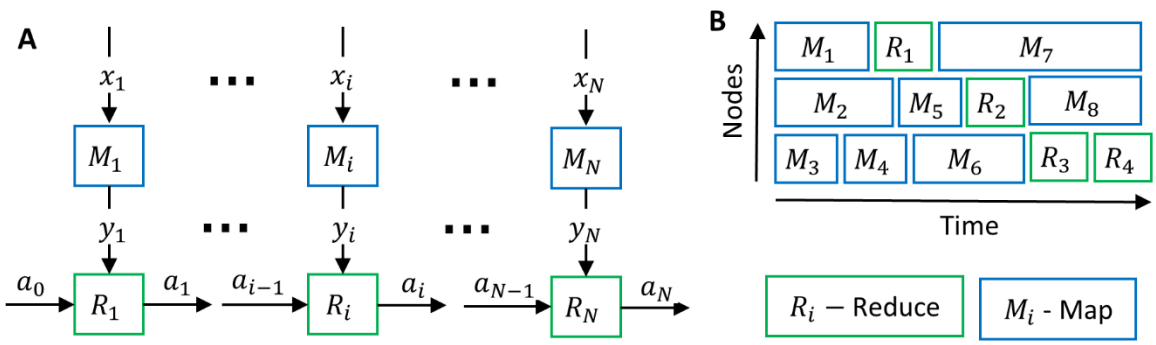



**Figure 2.3:** Schematic of the distributed computing framework used to process the incoming data at high speeds. **A.** Input data is divided into $N$ blocks $x_1$ to $x_N$. The 'map' computations $M_i$ are used to compute $y_i$ from $x_i$ in parallel. The 'reduce' computations $R_i$ are used to aggregate the outputs $y_i$ by serially threading a state $a_i$ through the outputs in order. **B.** Order of execution of the map and reduce computations in a 3-node cluster. The width of the boxes represent the duration of each computation. Map computations can happen in any order. Reduce computation $R_i$ is executed in the same node as $M_i$ and $R_{i+1}$ can only be started after $R_i$ has completed.

**SPIKE SORTING TO TRACK SINGLE UNITS OVER TIME**

A crucial data processing step (called 'spike sorting' in the literature[73–80]), necessary for relating extracellularly recorded single unit activity to behavior, is extracting the spike times of all the

neurons that can be reliably identified from the raw data in a given recording. Typically, the signal measured by an extracellular electrode – the fluctuations in the electrical potential at the tip of the electrode - contains contributions from the firing of hundreds of neurons in its vicinity scaled by the distance of each neuron from the electrode tip[81]. Often, only a few neurons are close enough to the electrode to elicit reliably large fluctuations in the measured voltage to enable separating those instances from fluctuations due to the firing of the hundreds of other nearby neurons and due to thermal noise.

However, identifying and tracking single neurons over the time-course of multiple hours, days and often weeks is an extremely difficult signal processing problem[75,76]. Since we are the first in the field to record single units continuously for weeks and wish to do this on a large scale, we found that existing spike sorting algorithms were not well adapted to this scenario and fail for two main reasons. First, spike sorting, even for short hour-long recording sessions, is an inherently hard problem (in the same sense as many problems in artificial intelligence/machine learning like vision or the cocktail party problem) due to the noisy nature of the data. As such, most commonly used approaches are only semi-automatic in nature with a lot of time consuming manual steps[75,76]. While a semi-manual approach towards spike sorting is not a severe bottleneck for hour long recording sessions it doesn't scale at all to 24x7 recordings. Second, due to relative motion between the electrode array and the brain, and possibly gliosis following implant, the spike waveforms associated with single units gradually change over the time-course of hours and days often disappearing into noise or emerging from it[51].

*Overview of the full multi-step spike-sorting algorithm*

We developed a new speedy algorithm that largely eliminates manual steps from spike sorting and is able to track single units over long periods of time (Figure 2.4). Our approach uses a combination of super-paramagnetic clustering[76,82] and integer linear programming to isolate single units and track them despite continuously changing spike waveforms. The approach is inspired by the segmentation fusion algorithm[83,84] proposed for connectomics, the 3D reconstruction of neural processes from a

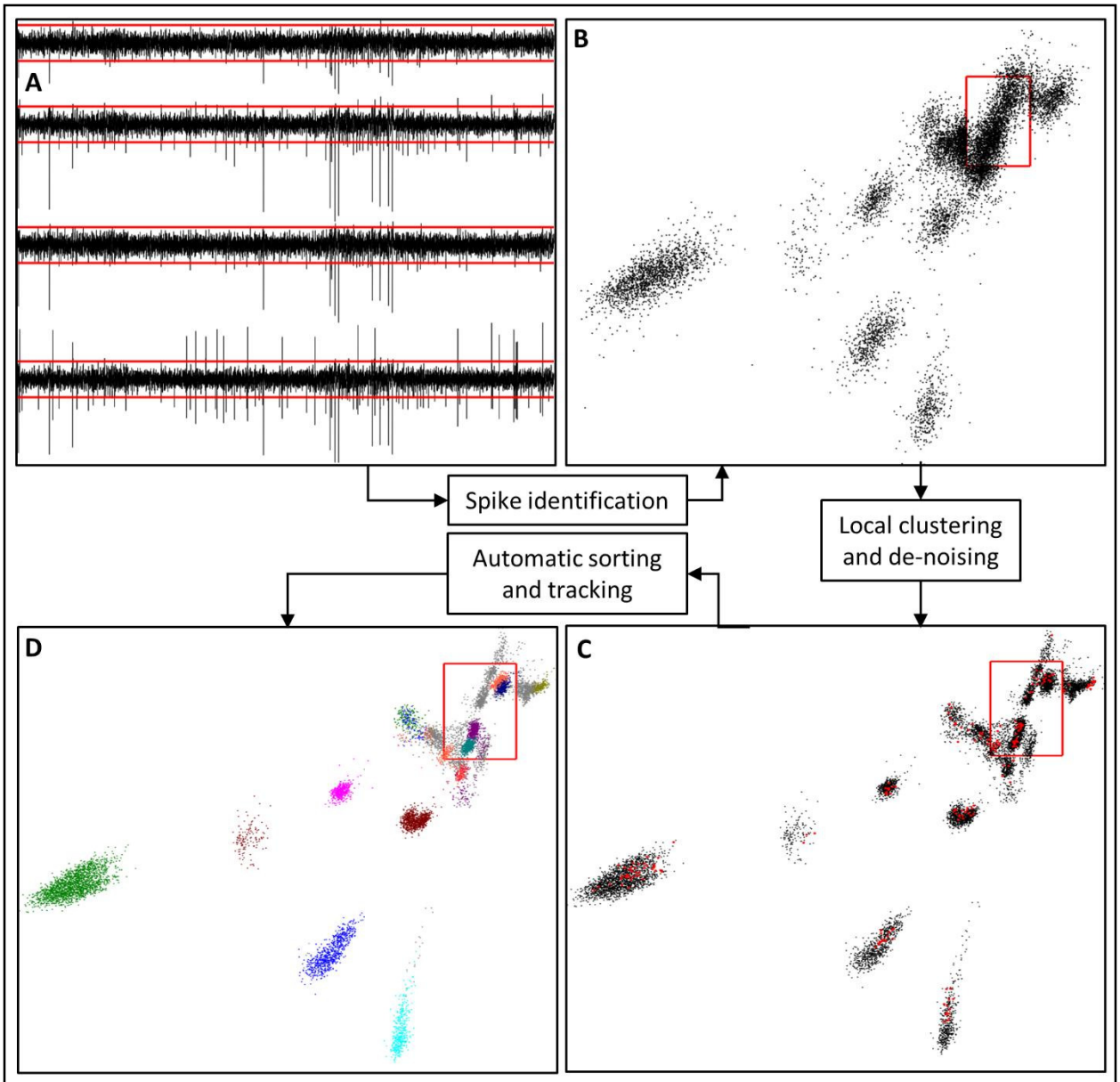



**Figure 2.4:** Steps for spike sorting of continuous long-term neural recordings. Spikes are identified from filtered raw data (panel **A**) by detecting large excursions from baseline (see Figures 2.5 and 2.6). Identified spike waveforms (panel **B**) are locally clustered and each cluster is replaced with its centroid resulting in de-noising of the spike waveforms and large reductions in the volume to data (see Figures 2.7 and 2.8). The averaged spike waveforms (panel **C**) are then separated into distinct units (panel **D**) and tracked over time (see Figures 2.9 and 2.10).

stack of 2D electron microscope images. The slowest step in the algorithm is currently approximately 3 times faster than the speed of data acquisition (referred to as 'real-time') and hence can be run on-the-fly on incoming neural data in our dual hex-core Xeon setup. Since the slowest step is CPU limited and

since the algorithm is trivially parallelizable it can easily be sped up even further. The entire processing pipeline contains only a few free parameters that control relatively well understood tradeoffs. We describe this pipeline below with parameters and examples chosen from a week-long recording in the motor cortex of a rat with 16 tetrodes.

Our algorithm has 3 main steps (Figure 2.4). First, spikes are identified in the raw data, then these spikes are locally clustered to produce a de-noised version of the original dataset. Finally single units are found and tracked from this de-noised dataset using integer linear programming, a type of constrained linear optimization algorithm.

**Spike identification**

*Overview of the spike identification step*

The first step in the spike sorting pipeline is to extract spike snippets, i.e. the millisecond long fluctuations in the measured signal presumed to be due to the firing of a neuron in the vicinity of the electrode, from the raw data. Mathematically, given the raw data for each channel over time ($s_{ch}(t), 1 \leq ch \leq 64$), and a grouping of the 64 individual electrodes into 16 tetrodes (for instance tetrode 1 might correspond to channels 1-4, tetrode 2 to channels 5-8, etc.), this corresponds to extracting a sequence of spike snippets - spike $i$ from tetrode $j$ is defined by the time of the spike, $st_i^j$, and its waveform, $\boldsymbol{x}_i^j$. The waveform contains 64 samples (2ms at a sampling rate of 30 kHz) from each channel of the tetrode concatenated together. Therefore, the waveform is a 256-dimensional vector.

*Details of the spike identification algorithm*

A schematic of this process is shown in Figure 2.5. To extract spike snippets the signal from each channel $s_{ch}(t)$ is partitioned into 15 second blocks with an additional 100 ms tacked onto each end of the block to account for edge effects in filtering. Then, for each block of each channel, the raw data is filtered with a 4$^{th}$ order elliptical band-pass filter (cut-off frequencies 400 Hz and 7500 Hz) first in the forward then the reverse direction to preserve accurate spike times and spike waveforms. Then, for each sample in

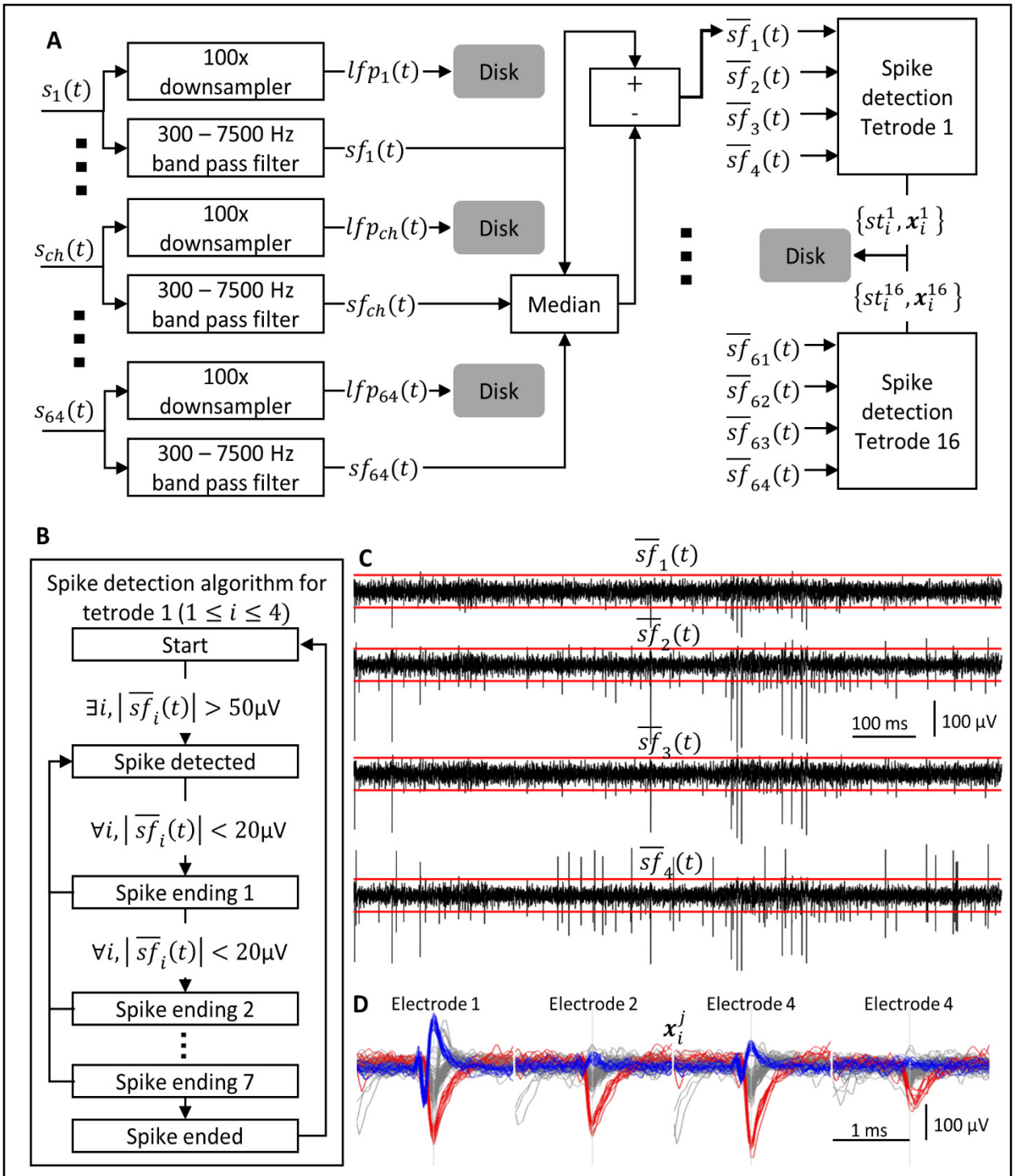



**Figure 2.5:** Algorithm for identifying spikes from the raw data. **A.** Each input channel $s_{ch}$ is split into two streams, one containing the low frequency components $lfp_{ch}$ and one containing the high frequency ones $sf_{ch}$. The median of $sf_{ch}$ across all channels is subtracted from each channel resulting in $\overline{sf}_{ch}(t)$. Spike times $st_i^j$ and spike waveforms $\boldsymbol{x}_i^j$ from each tetrode are then extracted. The LFPs and spikes extracted from the raw data is saved to disk resulting in a 5-10x 'compression' of the raw data. **B.** State machine for detecting spikes. If the absolute value of the filtered signal exceeds 50µV in

**Figure 2.5 (Continued):** any channel of a tetrode then a spike is 'detected'. The spike is considered to have 'ended' if all channels remain within 20 μV for 8 consecutive samples. **C.** Example 1 s long traces from a tetrode. The red lines mark the ±50μV spike detection threshold. **D.** 2 ms wide spike snippets (64 samples) extracted from data in C. Snippets from all 4 electrodes detected using the state machine of **B** are aligned to the peak of the spike waveform and concatenated to produce the 256 sample spike waveforms $\boldsymbol{x}_i^j$.

each block, the median across all channels is subtracted from every channel to eliminate common mode noise. This greatly reduces artifacts in the recording that arise from the animal chewing food or banging its head against the walls of the behavior box. It's important to use the median and not the mean because the former is a much more robust estimator of the common mode noise and is largely unaffected by neural firing. Finally, a threshold crossing algorithm in the form of a state-machine is used to detect spikes independently for each tetrode (Figure 2.5B-2.5D). A spike is defined as an event where the absolute value of the median-subtracted band-pass filtered signal exceeds a certain threshold. In our recordings, we use a threshold of 50μV which corresponds to about 7 times the median absolute deviation of the signal. After detecting a threshold crossing, we find the sample that corresponds to the local maximum of the event. This is defined as the maximum of the absolute value across all channels of the tetrode until the signal in all channels returns to baseline (20μV) for at least 8 consecutive samples. A 2ms (64 sample) snippet of the signal centered on the local maximum is then extracted from all channels. Therefore, each putative spike in a tetrode recording is characterized by the time of the local maximum and a 256 dimensional vector (64 samples x 4 channels, Figure 2.5D).

*Computational efficiency and data storage requirements*

Each 15 second block of each tetrode can be processed in parallel. However, since the number of spikes in any given 15 second block is not known in advance, the extracted spike snippets must be serially written to disk. Efficiently utilizing all the cores of the CPUs and simultaneously queuing disk read/write operations asynchronously is essential to keeping this step faster than real-time. For instance, a naïve

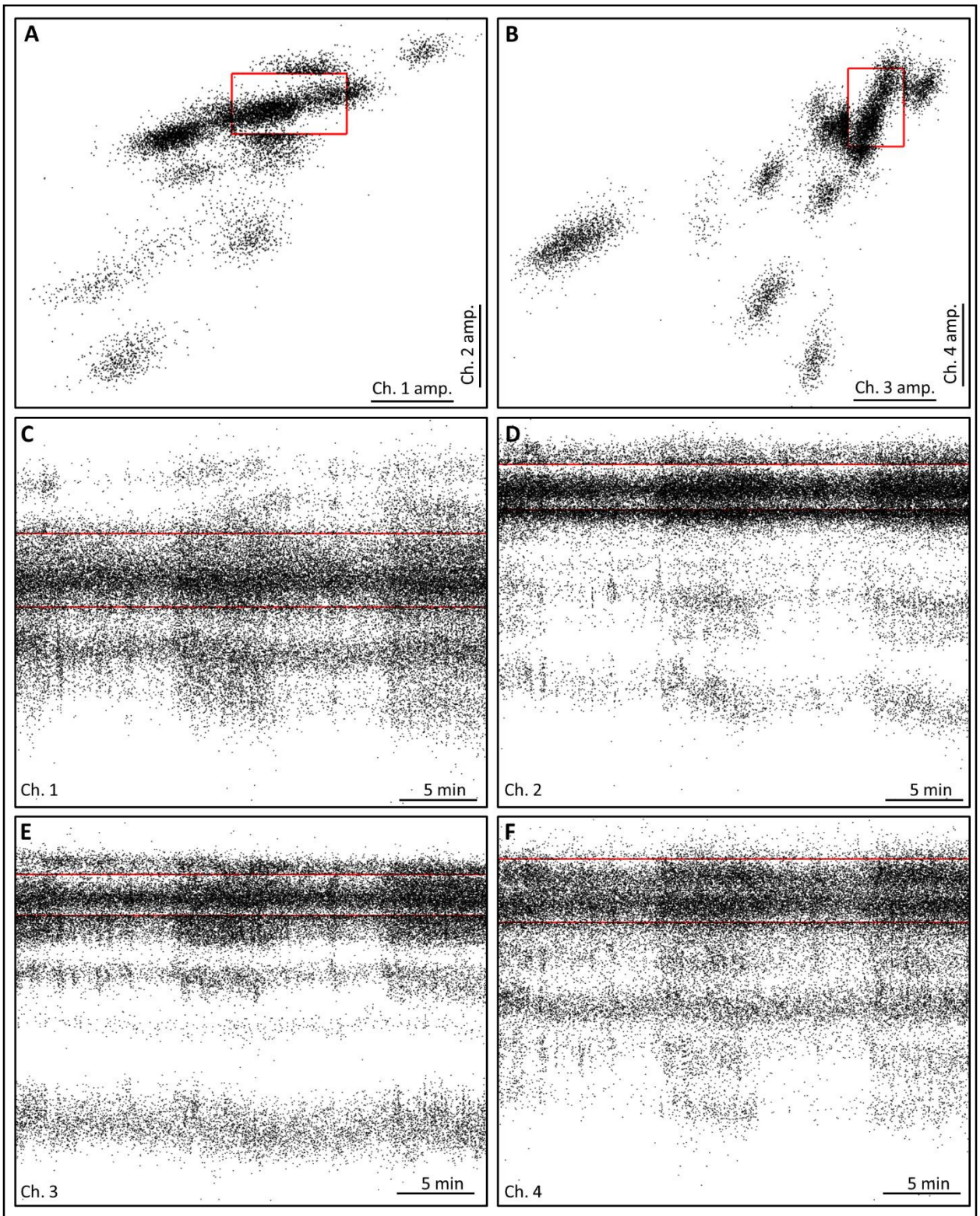



**Figure 2.6:** Different views of the set of raw spike waveforms $\boldsymbol{x}_i^j$**. A and B.** Two-dimensional projection of a set of 30,000 spike waveforms spanning 100 seconds. In panel **A** (**B**), for each spike waveform, the amplitude of the waveform on Channel 1 (3) is plotted against the amplitude of the waveform on

**Figure 2.6 (Continued):** Channel 2 (4). **C-F.** Scatter plot of spike amplitude over time for 50,000 spike waveforms spanning 30 minutes. Only every tenth spike is shown. The red boxes in **A** and **B**, and the red lines in **C-F** enclose a region encompassing $\pm 50\mu V$.

MATLAB implementation of this algorithm runs 2-3 times slower than real-time. We used our novel map-reduce framework to efficiently implement this algorithm. In our storage server, the filtering/spike detection step runs 15 times faster than real-time. After extracting the spike snippets and the local field potentials (LFP – the low frequency component of the raw voltage traces), the raw data can be deleted. This results in a 5-10x reduction in storage space requirements. To extract LFPs, we downsample the raw data 100-fold (from 30Khz to 300Hz) by two applications of a $4^{th}$ order 5-fold decimating Chebychev filter followed by a single application of a $4^{th}$ order 4-fold decimating Chebychev filter (Figure 2.5A).

A typical week-long recording from the motor cortex of rats with 16 tetrodes results in over a billion putative spikes. While most putative spikes are low amplitude and probably inseparable from noise (Figure 2.6), the spike detection threshold cannot be substantially increased without losing many cleanly isolatable single units. Assigning these billion putative spikes to clusters corresponding to single units as these clusters move around in the 256 dimensional spike waveform space in a largely automated fashion using a speedy algorithm is critical to successfully using the full potential of continuous 24x7 neural recordings.

*Sample output of the spike identification step*

Figure 2.6 shows a small subset of spike waveforms recorded using a 16-tetrode array in the motor cortex of rats. Panels A and B show the 30,000 waveforms extracted from just 100 seconds of the raw data from a single tetrode. Note that while some clusters are relatively well separated from the rest, others don't seem to be. Furthermore, different clusters can have very different densities which results from different neurons having very different firing rates. Figure 2.6B has an example of a very low density cluster. Panels C - F of Figure 2.6 shows 30 minutes of data from the same tetrode. Panel E shows the

same low firing rate unit mentioned above. Therefore, the challenge of spike-sorting is to develop an algorithm that takes the noisy data of Figure 2.6 and clusters it into distinct groups.

Our solution for this problem takes as input the time-ordered sequence of all $N$ spike-time, spike-waveform pairs from a tetrode $\{st_i, \boldsymbol{x}_i\}_{i=1}^{N}, st_1 \leq st_2 \leq \cdots st_i \leq st_N$ and produces a classification of a subset of the spikes into a set of well isolated units. Each single unit is defined by a set of indices $\{i_j\}_{j=1}^{M}, 1 \leq i_j \leq N$ and is estimated to have emitted spikes at times $st_{i_1} \ldots st_{i_M}$. Spike sorting is done independently for each tetrode by first locally (in the temporal domain) clustering the spike waveforms to reduce the volume of data and to produce de-noised estimates of the spike waveforms. This is followed by sequence of steps to identify the same cluster over the entire dataset. Each step of the algorithm is detailed below.

**Local clustering and de-noising**

*Overview of local clustering and de-noising*

This step of the algorithm converts the sequence of all spike waveforms $\{\boldsymbol{x}_i\}_{i=1}^{N}$ from a tetrode to a sequence of averages of spike waveforms $\{\boldsymbol{y}_i\}_{i=1}^{M}$ with each averaging done over a set of approximately 100 raw spike waveforms ($100M \cong N$) with very similar shapes that are highly likely to be from the same unit. The output of this step of the algorithm is a partitioning of the $N$ spike waveforms into $M$ groups. $\boldsymbol{y}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} \boldsymbol{x}_{i_j}, \sum_{i=1}^{M} N_i = N$. Local clustering followed by averaging produces de-noised estimates of the spike waveform for each unit during each point in time. The goal in this step is not to reliably find all the spike waveforms associated with a single unit but to be reasonably certain that the waveforms being averaged over are similar enough to be from the same single unit. This results in a dataset of averaged spike waveforms that is about a hundred times smaller than the original dataset greatly aiding in speedily running the remaining half of the spike sorting algorithm and in visualizing the entire weeks-long dataset at once.

*Super-paramagnetic clustering*

Clustering is inherently a scale dependent problem, i.e. the 'right' number of clusters in a given dataset depends on the scale being considered. At a very coarse scale, all points can be considered to be members of a single cluster and at a very fine scale each point belongs to its own cluster. A formal, mathematically precise way of characterizing this tradeoff is to think of clustering as lossy compression[85]. The amount of loss is defined as the amount of information lost by replacing each point in a cluster with their 'average' and the compression comes from characterizing the entire dataset with just the cluster averages. One simple loss measure is the sum of squared distances between each point in a cluster and the cluster centroid, i.e. the within cluster variance. If each point is assigned its own unique cluster then the loss would be zero. Conversely, if all points were assigned the same cluster then the loss would simply be the variance of the entire set of points. For intermediate amount of loss between these two extremes, the fewest number of clusters, i.e. the largest amount of compression, with at most that much loss, can, in principle, be computed. Conversely for a given number of clusters, one can, compute the clustering that results in the smallest amount of loss.

We use the super-paramagnetic clustering (SPC) algorithm[82] in our spike sorting pipeline partly since it parametrizes the loss-compression tradeoff discussed above with the 'temperature' parameter of the algorithms. At low temperatures, the algorithm assigns all points to a single cluster. As the temperature parameter is increased new clusters appear until, at very high temperatures, each point is assigned its own cluster. Units with large spike waveforms or very distinctive spike shapes appear at relatively low temperatures. However, other units often appear at relatively high temperatures and clusters at higher temperatures often don't include spikes in the periphery of the cluster. In existing uses of this algorithm for spike sorting the 'right' temperature for each cluster is selected manually[76]. Often several clusters at a higher temperature need to be manually merged as they all correspond to the same single unit.

The SPC algorithm also requires a distance measure between pairs of points. In previous approaches to spike sorting, a small number of features (on the order of 10) are extracted from the full 256 dimensional dataset (using a dimensionality reduction technique like PCA or by choosing the 10 most non-normal wavelet coefficients in a Haar wavelet decomposition of the spike waveform) and the Euclidean distance between points in this new feature space is used as the distance measure for clustering[76]. In our experience the number of coefficients that are necessary to adequately capture the distinction between similar but well isolated units varies substantially depending on the number of units being recorded on a tetrode and the signal-to-noise ratio of the recording. We find that simply using the Euclidean distance in the raw 256 dimensional space avoids this problem and is nonetheless not computationally prohibitive.

*Motivation for an iterative multi-scale local clustering algorithm*

Two considerations determine the size of temporal window used for each batch of local clustering. First, SPC requires computing distances between every pair of points, making the algorithm quadratic in the number of points being clustered in one batch. In a typical desktop-class PC, a window size of 10,000 spikes runs at a speed 8 times lower than real-time (1,000 spike batches on the other hand run somewhat faster than real-time). Second, gradual changes in the spike waveform of a unit, as measured at the electrode tip over time, results in the space occupied by points belonging to a single cluster to increase with the size of the temporal window, which in turn decreases the separation between clusters. Both these considerations favor clustering up to 1000 spikes at a time.

However, different neuron types in many brain areas, including the motor cortex, have very different firing rates – the highest firing rate units are often firing at more than a 100 times more frequently than the low firing rate units[86,87]. Therefore, a 1000 spike window might contain just a few or even no spikes from these units. To solve this problem, we developed a multi-resolution approach for this local clustering step. Our solution is to identify clusters corresponding to units with high firing rates,

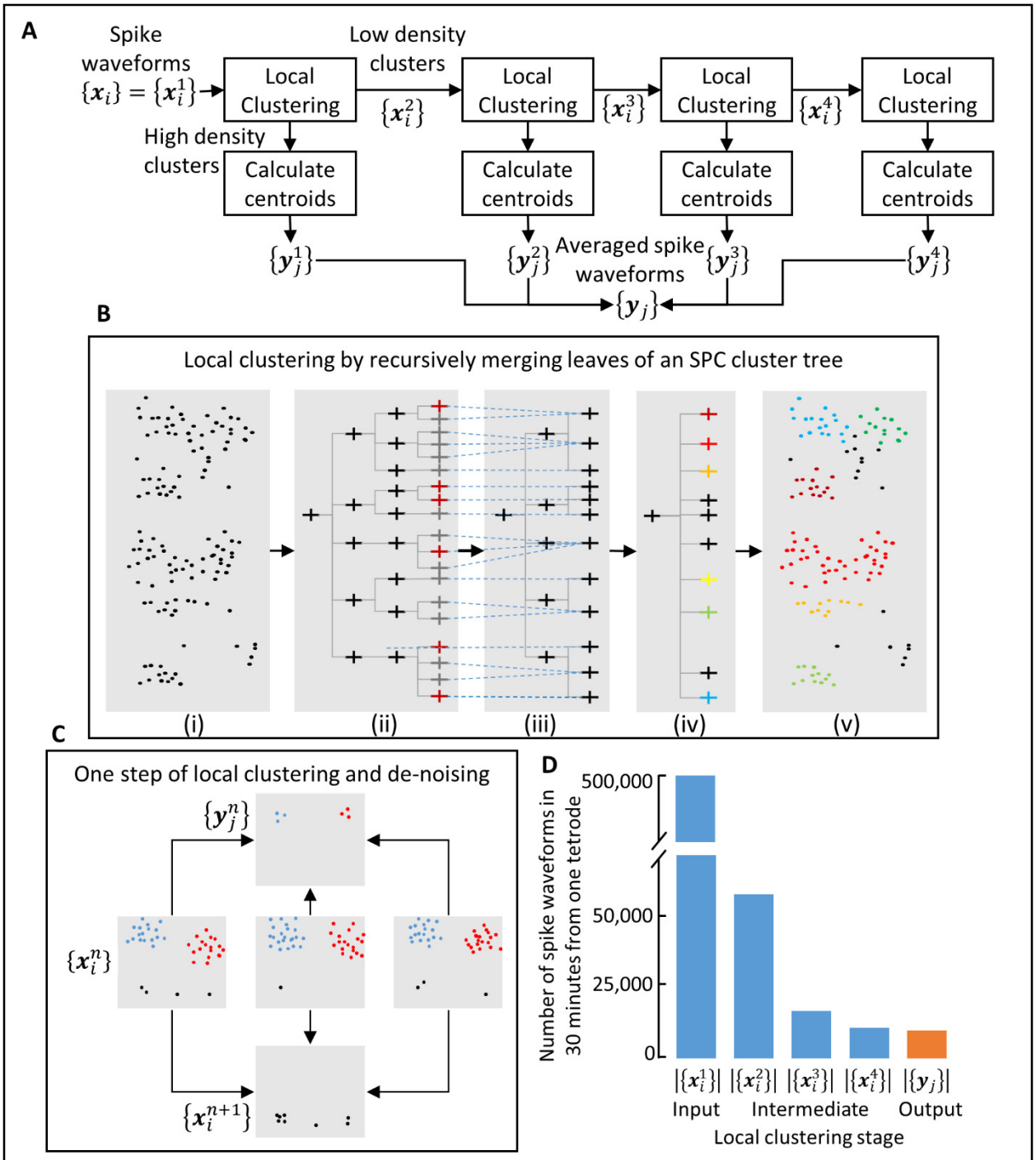



**Figure 2.7:** Algorithm for local clustering and de-noising. **A.** The raw spike waveforms $\{x_i\}$ are locally clustered and split into low- and high-density clusters (details in panels **B** and C). The spikes from low-density clusters are further split into two streams in the same manner 3 more times. The centroid of high density clusters from all 4 stages are pooled together to form the output $\{y_j\}$. **B.** Local clustering of each 1000 spike block. Super-paramagnetic clustering generates a cluster tree (ii) from the spike waveforms (i), the leaves of which are recursively merged (iii and iv) to generate a clustering of the 1000 points (v). The dotted blue lines show which leaves of the tree in (ii) are merged to produce the tree in (iii). The nodes marked red in (ii) correspond to 'distinct clusters', i.e. clusters that are very

**Figure 2.7 (Continued):** different from the parent nodes. The leaves of (iii) are similarly merged to produce the tree in (iv). The colored leaves correspond to high-density clusters, i.e. clusters with more than 15 points and the black leaves correspond to low-density clusters. **C.** Schematic illustrating splitting of spikes into low-density and high-density clusters. The set of input spike waveforms $\{\boldsymbol{x}_i^n\}$ is split into blocks of 1000 spikes (3 blocks shown in the figure) with each block split into low (colored black) and high density clusters (colored blue and red) using the procedure shown in panel **B**. The spikes from the low density clusters are pooled together to form $\{\boldsymbol{x}_i^{n+1}\}$. The centroid of the high density clusters form $\{\boldsymbol{y}_i^n\}$. **D.** Number of spike waveforms in a 30 minute period from one tetrode in various stages of the local clustering and de-noising algorithm.

remove them from the dataset, re-cluster the remaining spikes, and repeat this process iteratively (Figure 2.7).

*Details of the local clustering and de-noising step*

Step-by-step details of the algorithm for multi-resolution local clustering is described below and a schematic of the whole process is presented in Figure 2.7.

1. The set of all spike waveforms is partitioned into blocks of 1000 consecutive points. Therefore, the first block contains points $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{1000}\}$, the second block contains $\{\boldsymbol{x}_{1001}, \dots, \boldsymbol{x}_{2000}\}$ and so on.
2. An SPC cluster tree is generated for each block independently. This is computed by first clustering the set of 1000 points at a range of temperatures $T_i = 0.01i, 0 \leq i \leq 15$. This process assign a cluster label to each point at each temperature. This matrix of cluster labels is then converted to a tree where each node in the tree corresponds to a cluster at some temperature, i.e. a subset of the 1000 points. The root node of the tree (depth 0) corresponds to a single cluster containing all 1000 points. The children of the root node (depth 1 nodes) correspond to a partition of the set of 1000 points based on the cluster labels at temperature 0.01. For each node in depth 1, the children of that node (depth 2 nodes) correspond to a partition of the points associated with that node based on the cluster labels of those points at temperature 0.02. This is repeated for all temperatures to construct the full tree with depth equal to the number of temperature increments.
3. Each cluster tree is collapsed into a partition (a clustering) of the set of 1000 points (Figure 2.7B). The simplest technique for getting a partition from an SPC cluster tree is to use all the nodes at a fixed

depth, i.e. clustering at a fixed temperature. In practice, this approach suffers from major drawbacks. The lowest temperature at which a cluster first separates from its neighbors varies from unit-to-unit and depends on the signal-to-noise ratio of the spike waveform, how distinct the spike waveform of that unit is, etc. Also, when units appear at relatively high temperatures, the clusters corresponding to single units at those temperatures don't include many spikes at the periphery of those clusters. Therefore, instead of using a single temperature we recursively merge leaves of the cluster tree based on the loss-compression tradeoff discussed above to generate a partition. This is done by recursively collapsing the cluster tree one level at a time. Specifically,

a. For each leaf node in the cluster tree, the similarity between the leaf node and its parent is first calculated. Let $L = \{i_1, \dots, i_N\}$ be the leaf node which is specified by the indices of the subset of the 1000 points belonging that node. Similarly, let $P = \{j_1, \dots, j_M\}$ be the parent of the leaf node. Note that $L \subseteq P$. Let $\boldsymbol{l} = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{x}_{i_k}$ be the average spike waveform of the leaf node and $\boldsymbol{p}$ be the average spike waveform of its parent. Let $d_L = \sum_{k=1}^{N} ||\boldsymbol{x}_{\boldsymbol{i_k}} - \boldsymbol{l}||$ be the total distance of points in the leaf node from their average and $d_P = \sum_{k=1}^{N} ||\boldsymbol{x}_{\boldsymbol{i_k}} - \boldsymbol{p}||$ be the distance from the parent node. The difference $d_P - d_L = a_L$ measures how well the parent node approximates the leaf node, i.e. how much additional loss in incurred in approximating the points in the leaf node with the cluster corresponding to the parent node.

b. Let $L = \{L_i\}$ be the set of all N leaf nodes sharing the same parent node $P$. The set of leaf nodes that are poorly approximated by their parent (the well-isolated nodes $I$) are considered distinct clusters. $I \subseteq L$, where $L_i \in I$ if $a_{L_i} > a$. This encodes the intuition that if a cluster at a given temperature splits into multiple sub-clusters at the next higher temperature that are however each quite similar to the parent cluster then treating each of these sub-clusters as distinct clusters is inappropriate. The parameter $a$ provides an intuitive tradeoff between missing distinct clusters that appear at high temperatures and classifying spikes in the

periphery of a single cluster into multiple distinct clusters. Let $M$ be the number of elements in $I$. If any of the remaining $N - M$ nodes are well approximated by one of the well-isolated nodes then they are merged together. For $L_i \in L/I$, if $\min_{L_j \in I} d_{L_j} - d_{L_i} < a$, i.e. if node $L_j$ approximates node $L_i$ well then they are merged. Merging a set of nodes corresponds to creating a node containing all the points from each of the nodes being merged. This yields a set of augmented well-isolated nodes. Any remaining nodes, i.e. non-well-isolated nodes that are also not well-approximated by any of the well isolated nodes are merged with each other. Therefore, this step results in converting the set of N leaf nodes sharing a parent into a set of $M$or $M + 1$ nodes formed by merging some of them together.

c. A depth $D$ tree is converted into a depth $D - 1$ tree by replacing all the leaf nodes and their parents with the merged nodes derived in the previous step.

d. Step a – c are repeated recursively until the tree is of depth 1. The leaf nodes of this tree which are typically vastly fewer in number than the total number of leaf nodes of the original tree correspond to a partition of the set of 1000 points of each block.

4. The centroid of each cluster from the previous step containing at least 15 points contributes one element to the output of the local clustering step, the set of averaged spike waveforms $\{\boldsymbol{y}_i\}$ (Figure 2.7C). In our sample motor cortex dataset, clusters with at least 15 spikes in a 1000 spike window corresponds to firing rate of 5 Hz or greater. The points belonging to the remaining clusters, i.e. ones with fewer than 15 points, are all pooled together, ordered by their spike time and become the new $\{\boldsymbol{x}_i\}$. The number of spikes in this new subset is approximately 10% of the original. Steps 1-4 are used to locally cluster this new subset of spikes and produce a second set of averaged spike waveforms $\{\boldsymbol{y}_i\}$. This process of re-clustering the low-density clusters is repeated two more times. The averaged spike waveforms from all four scales are then grouped together to form the full set $\{\boldsymbol{y}_i\}_{i=1}^{M}$ and ordered by the median spike time of set of spikes that were averaged to generate

each $\boldsymbol{y}_i$. This process results in an assignment of over 98% of the original set of $N$spikes to a cluster with at least 15 spikes in one of the 4 scales. The firing rate of units in clusters with at least 15 spikes at the fourth scale is about 0.01Hz in the sample dataset.

*Sample output of the local clustering and de-noising step*

Figure 2.8 shows the output of this step of the algorithm on the sample dataset shown in Figure 2.6. Compared to Figure 2.6 the data is much less noisy and the clusters are much more compact. Furthermore, the volume of data has been massively reduced (Figure 2.7D). The original set of 500,000 spikes spanning 30 minutes is reduced to 10,000 average spike waveforms at the end of this step.

**Automated sorting and tracking of de-noised spike waveforms**

This step takes the sequence of averaged spike waveforms $\{\boldsymbol{y}_i\}_{i=1}^{M}$ computed in the previous step and generates an automatic spike sorting, i.e. identification of subsets of $\{\boldsymbol{y}_i\}$ that correspond to the same single unit. Loosely, a subset of $\{\boldsymbol{y}_i\}$ is considered to be the same single unit if distances between $\boldsymbol{y}_i$ and $\boldsymbol{y}_{i+1}$ are sufficiently small for the entire subset. As in the previous step, the set $\{\boldsymbol{y}_i\}_{i=1}^{M}$ is first partitioned into blocks of 1000 consecutive points and an SPC cluster tree is independently computed for each block, this time for the temperature range $T_i = 0.01i, 0 \leq i \leq 10$. Then, we use a binary linear programming algorithm inspired by a computer vision problem called segmentation fusion to identify the nodes in each cluster tree that correspond to the same single unit.

*Motivation for the use of binary linear programming to track units over time*

Tracking multiple single units over time from a sequence of cluster trees requires first selecting a subset of the nodes of each cluster tree that correspond to distinct units, followed by matching nodes from adjacent clusters trees that correspond to the same unit. Doing these steps manually is infeasible because of the volume of the data. In our sample motor-cortex dataset, the local-clustering step results in a total of 10 million averaged spikes from the original set of 100 billion spikes. This produces a set of 10,000 cluster trees making manual tracking impossibly labor intensive. In attempting to devise an

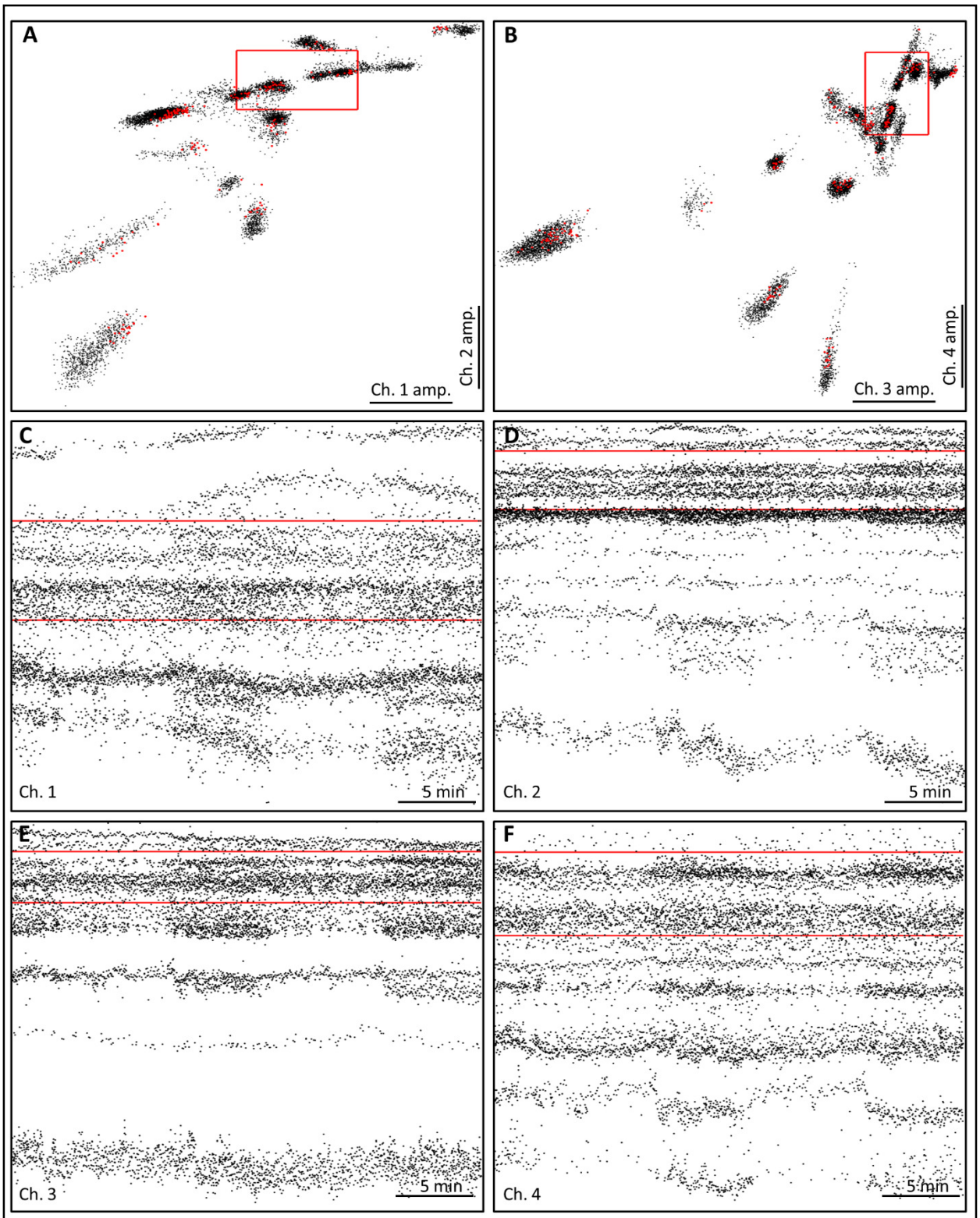


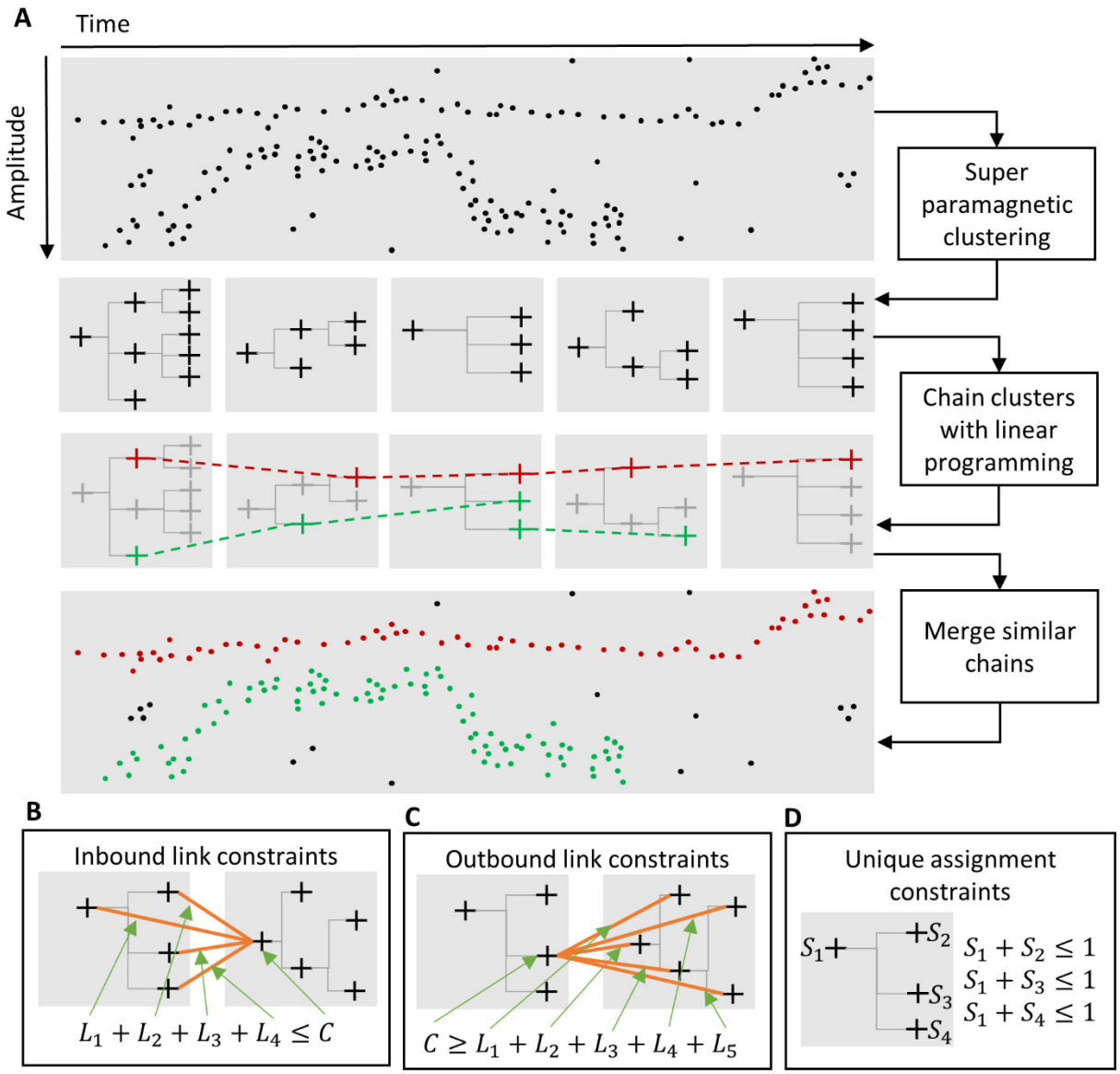

**Figure 2.8:** The set of 9643 averaged spike waveforms $\{\boldsymbol{y}_j\}$ that form the output of the local clustering step (Figure 2.7) for the same 30 minute long dataset in Figure 2.6 plotted in the same manner. The red dots in panels **A** and **B** highlights the subset of the 9643 points corresponding to the output resulting from the 100 second long 10,000 spike dataset show in Figure 2.6A and 2.6B.

automated algorithm for tracking single units over time, we discovered the segmentation fusion algorithm, a proposed solution to a mathematically analogous problem in the field of connectomics.

Segmentation fusion was invented to solve the problem of reconstructing the full 3D structure of the axons, dendrites and soma present in a volume of neuropil from a stack of 2D electron microscopy sections[83,84]. A cluster tree is analogous to a multi-resolution segmentation of the 2D image. Then identifying nodes across cluster trees that correspond to the same single unit is analogous to identifying the same segment across 2D sections as a neurite courses through the neuropil.

Before describing the details of the linear program, simpler approaches that were tried but failed to yield satisfactory results are discussed. The simplest approach is a greedy algorithm that starts with a seed cluster. A cluster corresponding to a well isolated unit from the first cluster tree is initially manually identified. Then the cluster 'most similar' to this seed cluster in the next cluster tree is identified. Then this newly identified cluster is used as the seed cluster to find the most similar cluster in the next cluster tree. This process is repeated throughout the entire dataset (or until the most similar cluster is nonetheless sufficiently different) and yields a sequence of nodes, one per cluster tree, that ought to correspond to the same single unit as the manually chosen seed cluster. However this algorithm fails in a number of ways. First, a simple similarity measure like the Euclidean distance between centroids of the two clusters fails to select the 'right temperature'. In general, the right temperature for a given cluster is the lowest temperature at which the cluster first appears. This deficiency can be largely eliminated by incorporating the temperature in the similarity measure as well. The more serious problem with the greedy approach is its lack of robustness. If an incorrect cluster is chosen at some step, perhaps because the data was particularly noisy during that time period, then all subsequent clusters in the sequence will also be incorrect.

A typical solution to the deficiencies of a greedy algorithm is to use dynamic programming. This is an optimization algorithm that takes into account the entire sequence of nodes rather than just the next




**Figures 2.9:** Algorithm for linking cluster trees to track single units over time. **A.** The output of the previous step (averaged spikes waveforms – see Figures 2.7 and 2.8) is split into a sequence of 1000 spike blocks and converted into a sequence of cluster trees (5 trees shown in the figure). A subset of all possible links between adjacent cluster trees is chosen by maximizing the total similarity between linked nodes subject to the constraints depicted in panels **B**, **C**, and **D**. The subset of chosen nodes and links are highlighted in color. Three sets of nodes connected by links, one in red and two in green, are shown. The two green chains are merged to produce a final sorting containing two units (red and green). **B.** The constraint shown ensures that none of the 4 incoming links ($L_1 - L_4$) are chosen if the node marked $C$ is not chosen. It also ensures that if $C$ is chosen, at most one of the incoming links are chosen. **C.** Same as **B** but for outgoing links. **D.** These constraints ensure that if a node is chosen then none of its parents or child nodes are.

node as in the greedy algorithm. Specifically, the algorithm picks the sequence of nodes amongst all

possible sequences that maximizes the sum of the similarity between adjacent nodes in the sequence. Despite the space of all possible sequences being extremely large, a computationally efficient algorithm for computing the optimal sequence exists. While this approach works quite well in a polytrode with just a single well isolated unit, it fails when multiple units are present since the algorithm only picks one node per cluster tree.

*Overview of the binary linear programing algorithm*

Using binary linear programming solves all these problem. At a high level, the algorithm finds a set of sequences of nodes from the sequence of cluster trees. Each sequence of nodes corresponds to a well isolated single unit. This is done by first enumerating all the nodes in all the cluster trees and all possible links between nodes in adjacent cluster trees. Then a constrained optimization problem is solved to find a subset of nodes and links that maximize a score that depends on the similarity between nodes represented by a link and the 'quality' of a node. This maximization is constrained to disallow assigning the same node to multiple single units and to ensure that if a link is selected in the final solution then so are the nodes on each side of the link.

*Details of the binary linear programming algorithm*

Each step of the binary linear programming algorithm is detailed below.

1. The sequence of cluster trees is grouped into blocks of 10 consecutive trees with an overlap of 5 cluster trees. Solving the binary linear program for blocks of larger than 10 trees is too computationally prohibitive.
2. The segmentation fusion algorithm is run independently for each block of 10 cluster trees (Figure 2.9). Let $\left\{\left\{C_j^i\right\}_{j=1}^{N_i}\right\}_{i=1}^{10}$ be the set of binary indicator variables representing all nodes in all 10 cluster trees where the cluster tree indexed by $i$ contains a total of $N_i$ nodes. The total number of nodes $N =$

$\sum_{i=1}^{10} N_i$. Let $\left\{\left\{L_{jk}^{i}\right\}_{j,k\,=\,1,1}^{N_i,N_{i+1,}}\right\}_{i\,=\,1}^{9}$ be the variables representing the set of all links between adjacent cluster trees. Link $L_{jk}^{i}$ connects clusters $C_j^i$ and $C_k^{i+1}$. The total number of links $M = \sum_{i=1}^{9} N_i N_{i+1}$. Solving the linear program requires choosing a $\{0,1\}$ value for each of the $N + M$ binary variables that maximizes the objective function $\sum_{ij} C_j^i\, \theta_j^i + \sum_{ijk} L_{jk}^i \left(\theta_{jk}^i - 0.02\right)$ . The objective function is a weighted linear sum of all the binary variables where the cluster weights $\theta_j^i$ represent the 'quality' of the cluster and the link weights $\theta_{jk}^i$ represent the similarity of the clusters joined by the link. The link weights are numbers in the range $(0,1)$. The threshold of 0.02 serves to give negative weight to links between sufficiently dissimilar clusters, effectively constraining the value of the variables representing those links to 0. This objective function is to be optimized subject to three sets of constraints. The first, $\sum_k\ L_{jk}^i \leq\ C_j^i$, enforces the constraint that if the node variable $C_j^i$ is assigned a value of 1 then out of all the outgoing links from the node $\left\{\ L_{jk}^i\right\}_{k=1}^{N_{i+1}}$, at most one is chosen (Figures 2.9C). Similarly, the second set of constraints , $\sum_j\ L_{jk}^i \leq\ C_k^{i+1}$, enforces the requirement that at most one incoming link to a node is chosen (Figure 2.9B). The third set of constraints enforces the requirement that for each of the 1000 points in a cluster tree at most one of the nodes containing that point is chosen (Figure 2.9D). This translates to inequalities $\sum_{k\,\in\,I_j} C_k^i \leq 1$ where the set of indices $I_j$ represents nodes in the path from the root of cluster tree $i$ to the $j^{th}$ leaf node of the cluster tree. Therefore, the total number of constraints of this type for each cluster tree is the number of leaf nodes in that cluster tree. The link weight $\theta_{jk}^i$ is the Euclidean distance between the average spike waveform of clusters $C_j^i$ and $C_k^{i+1}$ non-linearly scaled by a sigmoid function to fall in the range $(0,1)$. If $d$ is the distance then $\theta = \frac{a}{1+a}, \mathrm{a} = \exp\left(\frac{-(d-k)}{s}\right), \mathrm{s}\ =\ 0.005, \mathrm{k} = 0.03$. The parameter $s$ controls the steepness of the sigmoid and the parameter $k$ sets the distance $d$ at which $\theta = 0.5$. The cluster weight $\theta_j^i$ gives preference to clean well-isolated clusters, i.e. clusters that appear at low

temperatures and retain most of their points across a large temperature range. Let $N^{(0)}$ be the number of points in the cluster corresponding to $C_j^i$. Let $C_k^i$ be the largest cluster amongst the child nodes of $C_j^i$ and let $N^{(1)}$ be the number of points in $C_k^i$. Similarly let $N^{(2)}$ be the number of points in the largest cluster among the child nodes of $C_k^i$. Given the sequence of cluster sizes $N^{(0)}, N^{(1)}, \ldots, N^{(a)}$ where $N^{(a)}$ is the number of points in a leaf node of cluster tree, $\theta_j^i$ is defined as $N^{(0)}/\left(N^{(0)} + \cdots + N^{(a)}\right)$. This measure of cluster quality penalizes clusters that split into smaller clusters at higher temperatures and clusters that only appear at high temperatures.

3. The results of the previous step, i.e. the subset of the M links of each block that maximizes the objective function, are finally combined to produce a sorting that tracks single units over long time periods despite gradually changing waveforms. Links that are part of two instances of the segmentation fusion procedure due to the overlap mentioned in step 1 are only included in the final solution if both linear programs include them. The set of links chosen by the segmentation fusion algorithm are chained together to get long chains of clusters. For instance if links $L_{jk}^i$, $L_{kl}^{i+1}$, $L_{lm}^{i+2}$ are assigned values of 1 in the solution to the segmentation fusion linear program then all points in clusters $C_j^i, C_k^{i+1}, C_{jl}^{i+2}, C_m^{i+3}$ belong to the same chain and hence the same single unit. Each point that does not belong to any chain is assigned to the chain containing points most similar to it (as measured using the sigmoidal distance of step 2) as long as the similarity $\theta > 0.02$ (again the same threshold as used in step 2).

*Merging the chain of nodes and links identified by the binary linear programming algorithm*

Often, spike waveforms have multiple equal amplitude local extrema. Since the waveforms are aligned to the local extrema with the largest amplitude during the spike identification phase, different waveforms from the same unit can be aligned to different features of the waveform. This results in multiple chains for the same single unit since the Euclidean distance between waveforms aligned to

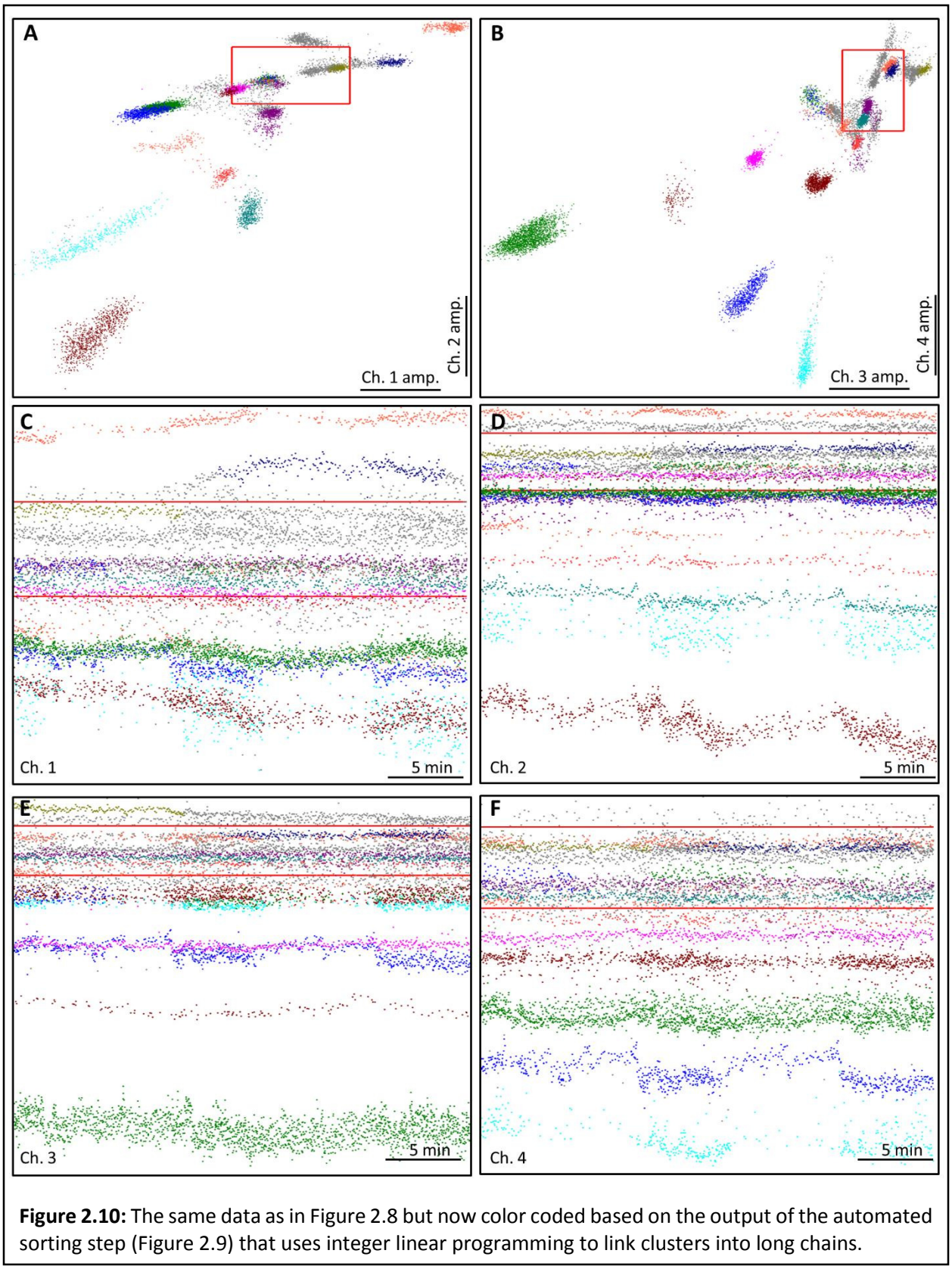



**Figure 2.10:** The same data as in Figure 2.8 but now color coded based on the output of the automated sorting step (Figure 2.9) that uses integer linear programming to link clusters into long chains.

different features is very large. This is remedied by merging chains that contain spikes from the same recording interval if the translation-invariant distance between the spike waveforms of the chains is sufficiently low in the overlap region. The translation invariant distance is computed by first calculating the distance between a pair of spike waveforms for a range of relative shifts between the pair and then taking the minimum of this set of distances. Overlapping chains with the smallest translation-invariant distance are first merged. This is done recursively until either no overlap between chains remains or overlapping chains have distinct spike waveforms and hence correspond to different simultaneously recorded single units.

*Sample output of the automated sorting and tracking step*

The output of this step of the algorithm on our sample 30 second dataset is shown in Figure 2.10. Because of the local clustering and de-noising step, the clusters are relatively far apart and hence can easily be separated from each other. Also, note that the cluster chaining algorithm successfully tracks the many simultaneously recorded units on this tetrode despite changing spike waveforms.

**Visualization and manual verification**

Visualizing the raw input data, intermediate results, and the final output of the spike-sorting algorithm described above is critical for verifying the quality of the results and understanding the tradeoffs encoded by the various parameters of the algorithm. However, the enormity of the dataset even for just a single tetrode makes this an exceedingly challenging problem. Existing data visualization libraries in a variety of programming environments like MATLAB, .NET or Python are simply incapable of doing this even in a reasonably powerful workstation. Therefore, we developed a GPU accelerated high-performance graphics engine and a suite of interactive data visualization tools on top of it to make this feasible. These include a GUI for visualizing spike amplitudes recorded in a tetrode over time as a scatterplot with the ability to interactively select a subset of the spikes to view their full waveforms (Figure

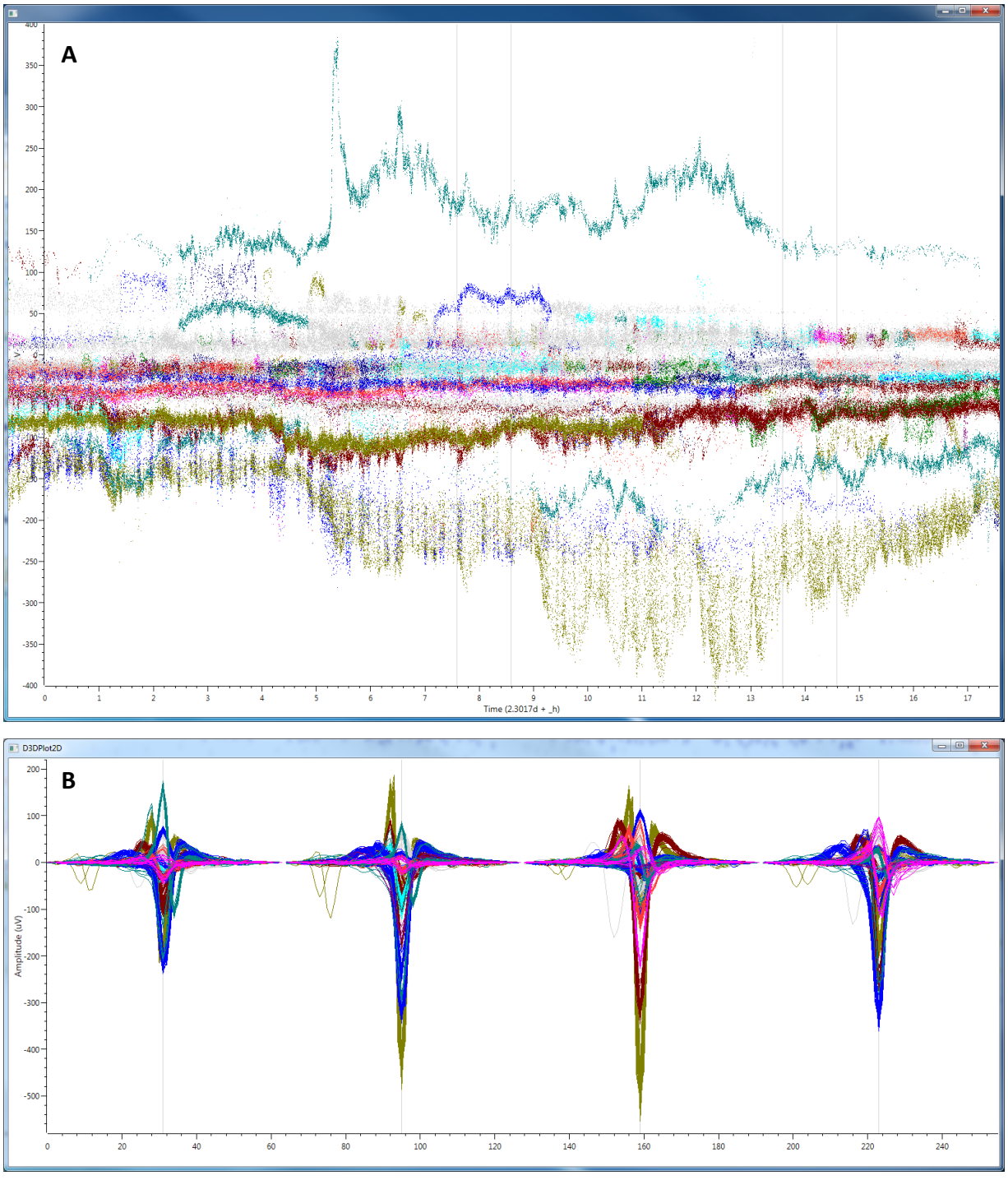



**Figure 2.11:** Interactive GUI to view spike waveforms. **A.** Spike feature viewer. Displays a 1-dimensional projection (here amplitude on channel 3 is shown) of spike waveforms over time. This snapshot shows 300,000 spikes over 18 hours**. B.** Spike waveform viewer. Displays the full spike waveform in all channels of a tetrode for subsets of spikes interactively selected from panel A. Both interfaces support real-time zooming, panning and interaction. Subsets of points (and waveforms) can be colored (here coloring is based on the output of the automated sorting pipeline).

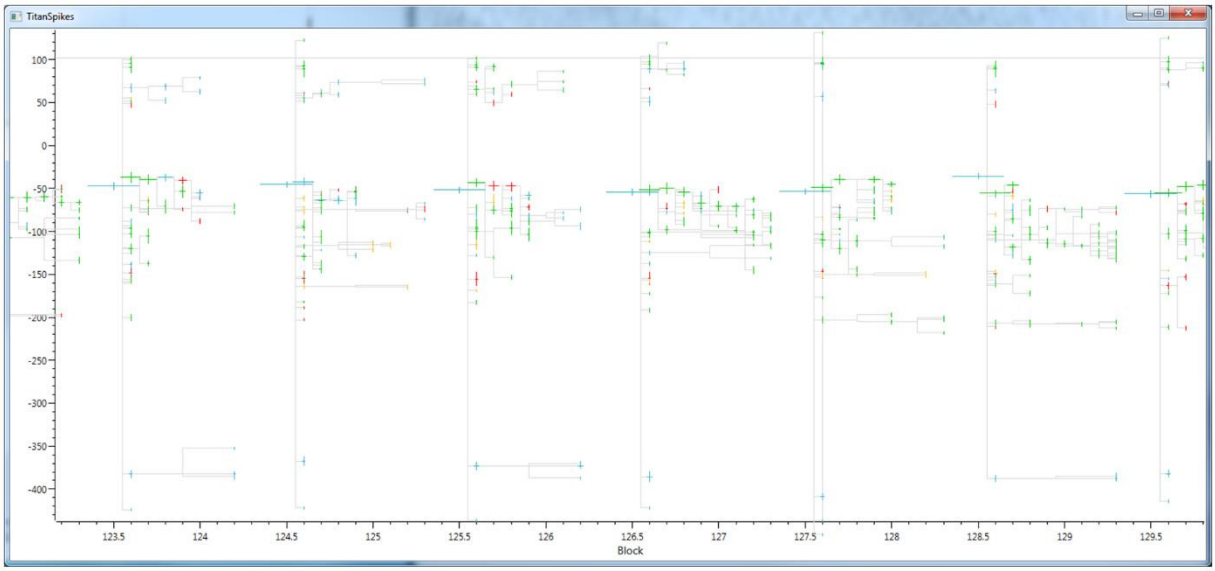



**Figure 2.12:** Interactive GUI to view cluster trees. This snapshot shows a sequence of 6 cluster trees. Each node of each tree is represented by a cross-hair. The width, height, position and color of the crosshairs encodes various features of the cluster represented by each node.

2.11). This GUI can be used to view the raw spike data (the output of the spike identification step) to get a quick estimate of the number of units, their typical shapes, and how stable the shape is over time. Also, it allows interactively computing an SPC cluster tree on selected subsets of spikes. This can be used to experiment with parameters of the SPC algorithm like the temperature range and the distance function. This GUI can also be used to view the final output of the spike sorting pipeline. In this mode, the GUI displays the amplitudes of average spike waveforms from the output of the local clustering and de-noising step over time. The entire dataset for a single tetrode containing over a million spike waveforms can be interactively viewed at once. All points belonging to the same single unit are colored the same. This view allows manually checking the quality of the clustering. A second GUI was developed to view the output of SPC clustering (Figure 2.12). In this interface, a sequence of trees with each node represented by a crosshair shaped marker is used to visualize the results of the clustering step with the ability to interactively view the full spike waveforms of all the spikes in any node of a cluster tree. The graphical attributes of the markers representing each node is used to encode various properties of a cluster like the

temperature (x-position), the amplitude of the spike waveform (y-position), which of the 4 electrodes of a tetrode has the largest amplitude (color), the stability of the cluster (vertical size of the marker), and the number of spikes in a cluster (the horizontal size of the marker).

### System Validation

The primary goal in developing the spike sorting pipeline described above was to eliminate as much of the manual steps as possible and to ensure that the automated algorithm is faster than real time and easily scalable to high channel-count electrode arrays in a large number of animals in a cost effective manner. We intended to solve the unique difficulties in spike sorting that arises specifically in the setting of continuous weeks-long recordings in awake animals rather than improving the state-of-the-art for short duration recordings. We wanted the algorithm to be able to track large amplitude units continuously for many days, despite substantial changes in their waveform during this period, completely automatically. For units with lower amplitudes and hence lower SNR the goal was to reduce the number of manual corrections necessary to a manageable number (approximately 1 day of manual work for a month of recordings). Since spike sorting is still a relatively subjective process, and since formal evaluation of the quality of spike sorting is very difficult (for reasons mentioned below), we focused on developing graphical tools to rapidly and interactively view the results of the sorting rather than devising more formal measures of sorting quality.

The ground truth in extracellular neural recordings, i.e. a grouping of putative spikes as identified by an amplitude threshold into clusters corresponding to single units, is usually unknown. Comparing different spike sorting methods is therefore subjective, and most labs rely heavily on the intuition of the researcher to make such comparisons. The most reliable way to get the ground truth is to perform simultaneous intra-cellular and extra-cellular recordings and use the ground truth from intra-cellular recordings to evaluate the performance of spike sorting from signals recorded by the extracellular

electrodes. However, intra-cellular recordings in-vivo is technically very difficult and hence very few such datasets exists[74,78]. Furthermore, current technology does not permit lengthy intra-cellular recordings – typical experiments record for less than an hour – and hence does not give a ground truth dataset for weeks-long recordings. Another approach towards getting approximate ground truth is to use polytrodes, i.e. bundles of electrodes that are very close to each other such that they all sample electrical activity from the same set of neurons. Occasionally, the spike amplitude of a unit as recorded in one electrode of the

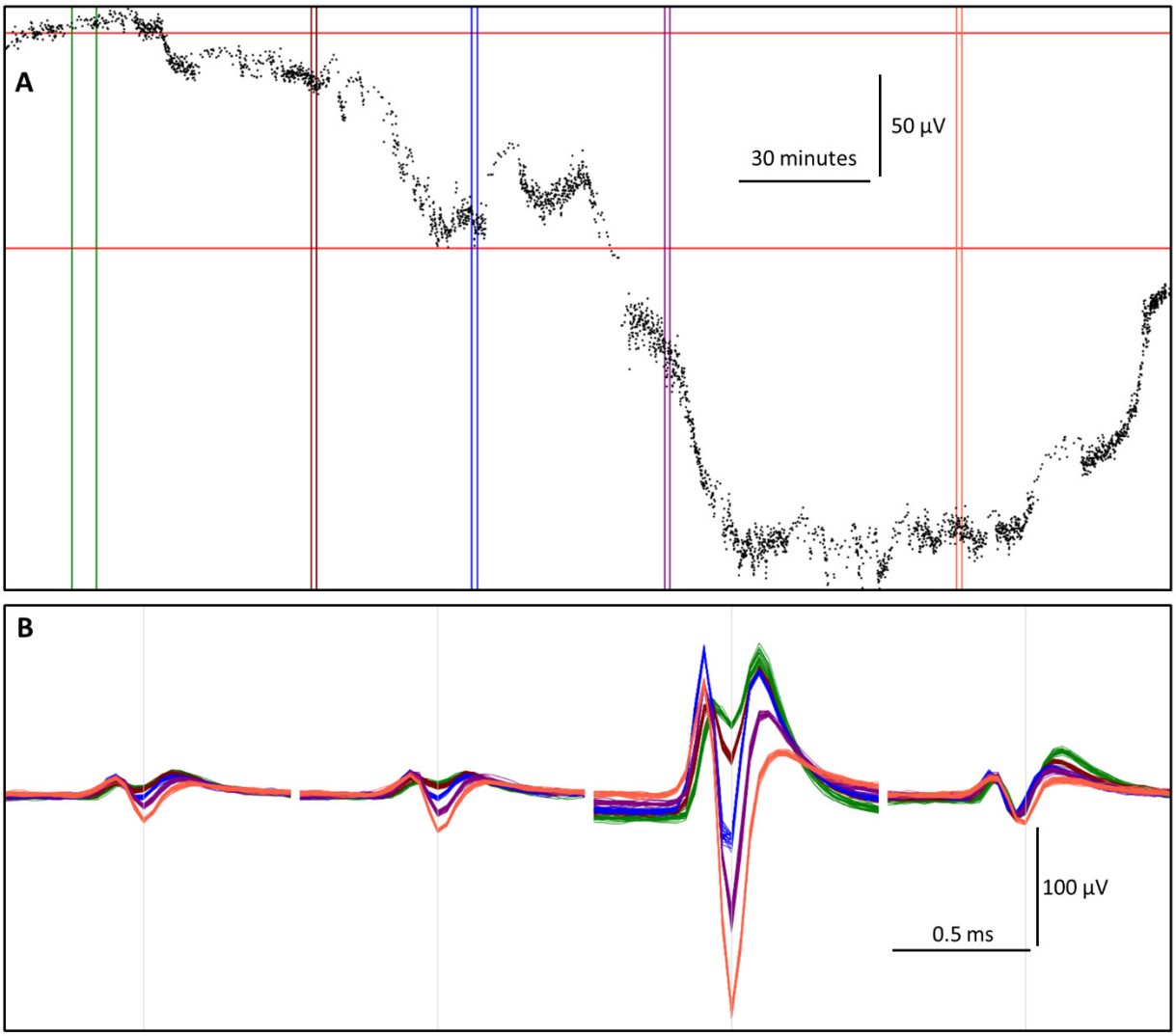



**Figure 2.13:** An example unit recorded for 4.4 hours with substantial but gradual changes in the spike waveform. **A.** Each point represents an average of 100 consecutive spike waveforms. The x-coordinate is time and the y-coordinate is the amplitude of the waveform on channel 3 aligned to the trough in the waveform. As before the red lines mark ±50µV. **B.** 25 consecutive spike waveforms from 5 time periods as indicated by the color coded vertical lines in panel A.

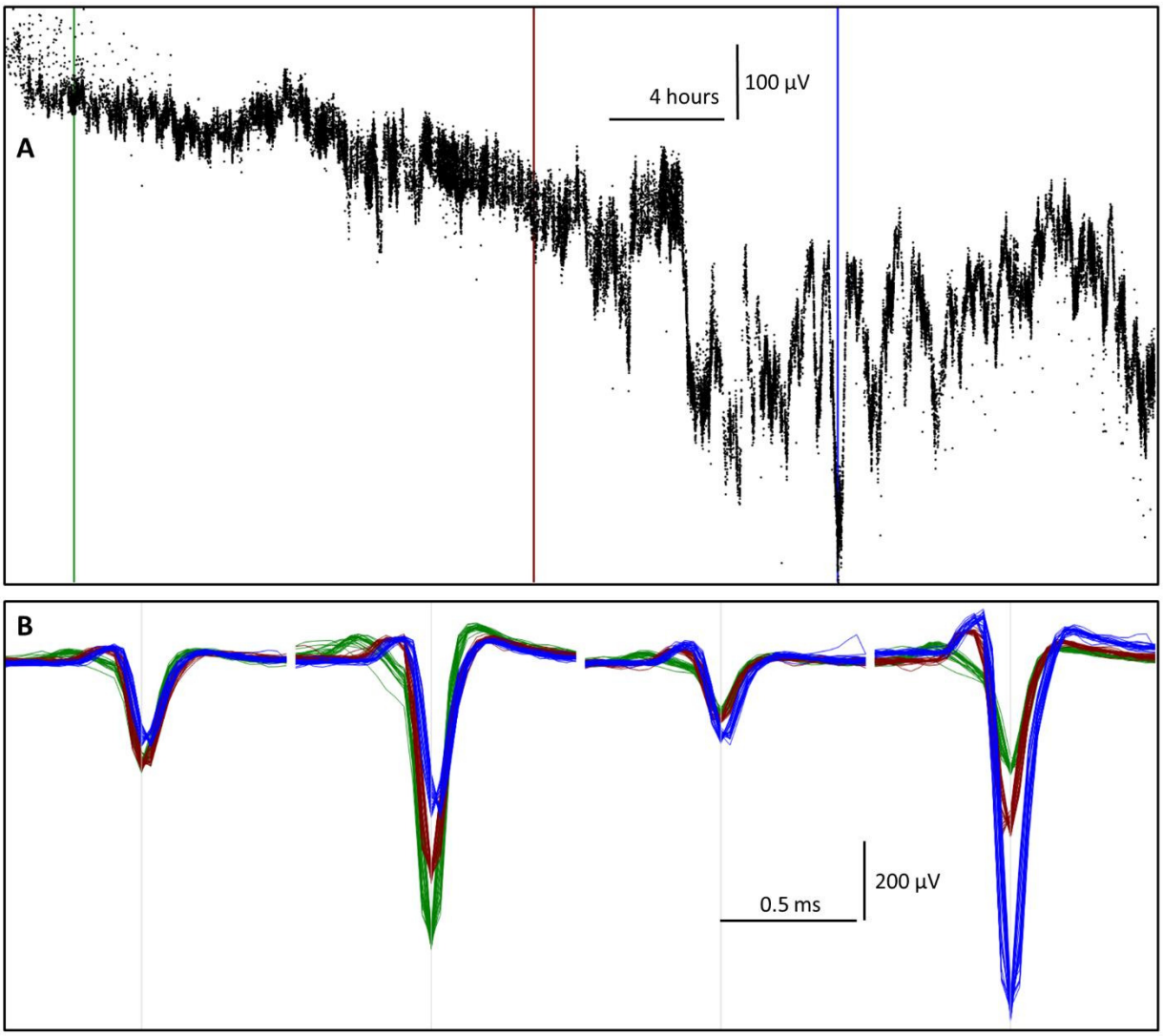



**Figure 2.14:** Similar to Figure 2.13. Depicts a large amplitude unit tracked for 40 hours.

bundle can be extremely large and easily separable from noise and other units by a simple amplitude based discriminator. This can then be used as the ground truth to evaluate the performance of spike sorting based on the remaining electrodes in the polytrode. This however requires, just the right set of circumstances and is dependent entirely on chance. A third technique for comparison of spike sorting methods is to generate synthetic datasets where the ground truth is exactly known[76]. This requires accurately modeling the statistical properties of real neural recordings. Given that we lack a detailed

understanding of the physical processes that result in changing spike waveforms in long-term recordings from behaving animals, this approach is also problematic.

Here, we show examples of the result of our spike sorting algorithm, on real data recorded from the motor cortex of rats (Figures 2.13, 2.14 and 2.15). The parameters of the algorithm were not separately fine-tuned for different tetrodes/animals/brain areas and hence our results could probably be improved upon. In the next chapter of this thesis we show that the activity of these neurons in relation to behavior remains remarkably stable over the time course of days and weeks providing yet another validation of our system.

The first example shown in Figure 2.13 shows a neuron whose spike waveform radically changes over the course of just 4 hours. At the beginning of the period depicted in Figures 2.13, the unit produces a waveform with three local extrema, two maxima separated by a minimum (green waveform in Figure 2.13). The entire waveform lies above the y-axis. Over the next few hours, perhaps due to relative motion between the measuring tetrode and this cell, the local minimum in the feature becomes increasingly less positive and eventually becomes highly negative ending up with a spike shape that looks much more like the typical unit. During this period of transition, the local extrema that is farthest away from the baseline changes. The chain merging step (sub-step 4 of the *'Automated sorting and tracking of de-noised spike waveforms'* step of the algorithm) then merges the differently aligned spike waveforms into one.

The second example shown in Figure 2.14 is much more common than the first example. Here too, the amplitude of the spike waveform changes from 200μV to 900μV over the course of 30 hours. In this example, however, the spike shape mostly transforms by scaling to different amplitudes. When the spike amplitude is large (> 500μV) the changes in the spike amplitude are also very large and quite rapid. We found this to be true of almost every single unit with a large amplitude.

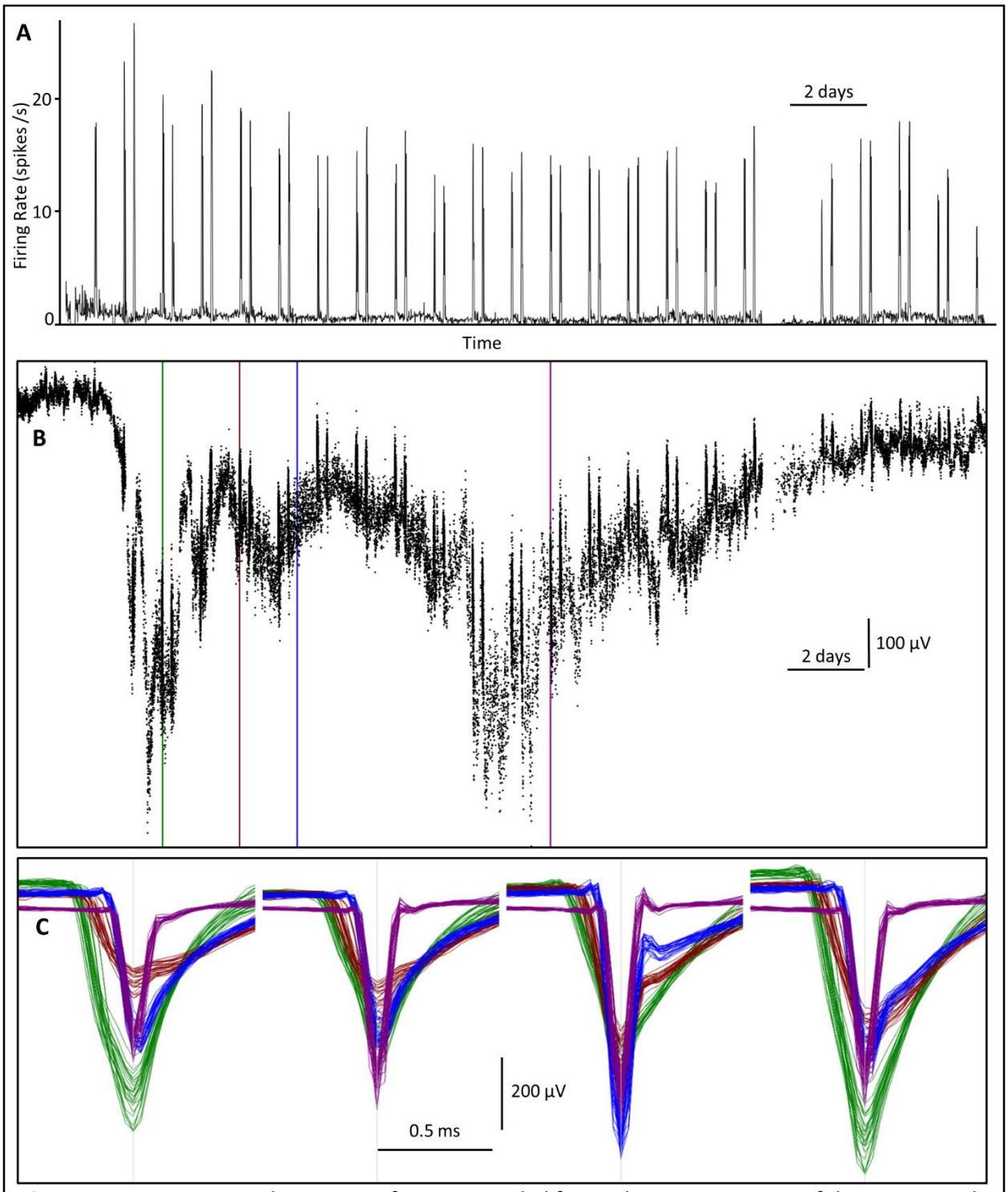



**Figure 2.15:** Depicts a 25 day section of a unit recorded for 41 days. **A.** Firing rate of the continuously recorded unit. The regularly spaced peaks in the firing rate correspond to two hour-long training sessions each day. **B and C.** Similar to Figures 2.13 and 2.14. Spike amplitude systematically decreases during the high firing rate mode corresponding to training sessions. Also spike waveform gradually narrows over days. **A** and **B** have the same timescale and span a total of 25 days. There is a short disruption in the recording at day 20.

The third example shows a 25 days section of a unit recorded for 41 days. This unit was recorded almost without interruption for the full 25 days. The spike waveform of this unit changed in a very peculiar manner and this phenomenon was not seen in any other unit. Our automated algorithm only required a few manual corrections (less than 5 minutes of manual work to visually browse through the entire dataset and merge chains that were erroneously split) to track this unit despite the radical changes in the waveform. The spike shape of this unit started out relatively broad and gradually became very narrow over time.

## Discussion

### *Summary of the system for continuous long-term neural recordings*

Successful deployment of a largely automated system for weeks-to-months-long continuous 24x7 neural recordings with high channel-count electrode arrays from behaving animals poses numerous engineering challenges. These include, creating a home-cage based automated training system, preventing animals from destroying the cable connecting the headstage to the data acquisition system and storing/processing the voluminous amounts of raw neural data that gets collected. Furthermore, to scale this system to dozens or hundreds of animals requires designing low-cost hardware and speedy software. The most challenging step however is automating the laborious and subjective spike sorting process despite the presence of large gradual changes in already noisy spike waveforms. Ideally, all the steps between implanting an electrode array and receiving a set of sorted single units and LFPs temporally aligned to high dimensional behavioral data would be completely hands off. While spike sorting still requires some amount of manual verification and correction, our end-to-end system substantially reduces the effort involved in making it feasible for a single researcher to manage the acquisition of neural and behavioral data from dozens of animals simultaneously.

In the previous chapter we described and characterized a general purpose system for training dozens of animals in their home cages with minimal human involvement. Here, we described an engineering solution for enabling continuous tethering of rodents for weeks that can be added to any home-cage (Figures 2.1). Using a system of linear slides, pulleys and counterweights our design allows recording neural data from behaving rats without interruption for weeks. Once wireless power transmission technologies become more mature, we anticipate a wireless telemetry system having several advantages over ours – particularly the ability to record in much larger environments than a cage. However, power requirements increase linearly with the number of channels simultaneously recorded and with the addition of accessory devices on the rat's head like a miniature camera and so we think wired neural recordings will remain an attractive option for some time. We also showed that continuous 24x7 tethering did not adversely impact the performance of animals in at least some tasks.

Continuously recording 64 channels of data sampled at 30 kHz requires 0.5 TB of storage per animal per day. We developed an ultra-low-cost storage solution with a price that is dominated by the spinning hard disks in the storage server (Figure 2.2). It is nonetheless highly reliable and provides very high bandwidth access to the data. This proved critical for exploratory data analysis which is often not very CPU intensive and hence IO limited. Furthermore, we parallelized virtually all steps of the standard neural data processing pipeline like filtering, spike detection, and clustering, and developed a new distributed computing framework to maximally use all available CPU and IO resources (Figures 2.3).

Since continuous recording for even just a week results in over 1 billion putative spikes and since the shape of spike waveforms from single units change dramatically over hours and days, existing spike sorting software was incapable of parsing this dataset. Even just the simple problem of filtering and detecting spikes based on amplitude thresholds requires carefully parallelizing the work load for faster than real-time performance. To make sorting the identified spikes computationally tractable we used a strategy of first dividing the full weeks-long dataset into many blocks of smaller subsets of the data,

followed by independently clustering each block, and finally by identifying the same unit across blocks. Choosing the set of parameters for clustering and identifying the same cluster over time in an ad hoc manner proved not to be generalizable. A clustering algorithm that is commonly used in spike-sorting, super-paramagnetic clustering, at a minimum requires picking the 'right temperature' for a cluster and often requires merging several clusters. Simple heuristics for automating this manual process did not generalize even for the same tetrode at different time points since the number of separable single units in a tetrode, their waveform, and the background noise characteristics varied substantially over time. Furthermore, successfully sorting low firing rate units from high firing rate ones also proved challenging.

Therefore we developed a novel multi-stage algorithm combining several passes of super-paramagnetic clustering followed by integer linear programming that is able to automatically produce a sorting of the full dataset (Figures 2.7 and 2.9). We intentionally set the parameters of the algorithm to err on the side of splitting one cluster over merging distinct ones which leaves the option of manually merging split clusters later. Since the correctness of spike sorting is often subjectively determined, we found it crucial to be able to rapidly visualize large amounts of raw data. To this end, we developed a high performance graphics engine for plotting millions of points and tens of thousands of spike waveforms interactively (Figures 2.11 and 2.12). This was indispensable in developing the spike sorting pipeline and in verifying and occasionally manually altering its output.

Several examples shown in Figures 2.13 – 2.15 show just how substantially spike waveforms can change over hours, days and weeks. At present, the spike-sorting algorithm that we have developed takes the raw data from the storage server and automatically, computes a conservative sorting of the full dataset. Then, a user can visualize the output of the sorting algorithm and rapidly verify its correctness and manually adjust the output when necessary. We found that it takes about 1 day to go over 1 month of data from 16 tetrodes implanted in one animal.

*Deficiencies of the system and avenues for improvements*

The algorithm is by no means fully optimized - further enhancements to the algorithm should reduce the amount of manual verification/correction necessary even more. Major avenues of improvement in the algorithm include a systematic exploration of the parameter space to identify the optimal parameters and a better merging algorithm that can intelligently adjust the threshold used for deciding whether two chains are indeed the same neuron in a context specific manner. The current merging algorithm simply merges overlapping chains with similar waveforms based on a simple globally constant threshold of similarity. A better algorithm would adjust this threshold based on the noise characteristics of the waveforms being considered.

Yet another improvement to the system would be to output not only a sorting of the set of input spikes but also a confidence value associated with each spike, i.e. how likely it was to be misclassified. This could be computed by combining a variety of metrics already available to the algorithm like the stability of a cluster as defined for the linear programming step of the algorithm and the distance between waveforms as defined for the merging step. Then a user interface could allow for checking the correctness of sorting of the spikes that were most likely to be misclassified.

## Attribution

Rajesh Poddar conceived, designed, and built the system and its component hardware, software and algorithms. Rajesh Poddar and Ashesh Dhawale did the recording experiments. Rajesh Poddar wrote the chapter.

# Neural representation of learned motor sequences in the motor cortex of rats

## Introduction

Many human behaviors, like speaking, typing, and dancing, consist of sequences of movements in a particular order and often with precise timing. Despite the ubiquity of movement sequences in our behavioral repertoire and the ease with which we acquire new ones, very little is known about their neural basis and correlates. How is the activity of neural circuits in relevant brain areas related to the dynamics of muscle movements that result in complex behaviors? How do neural circuits produce the neural network dynamics? How are they shaped to increasingly approximate desired action patterns? What is the structure of the neural representation of a learned motor sequence? Is the neural representation of a movement in a particular brain area specific to its sequential context? Is the first element of a sequence represented differently from the second? How precisely is neural activity correlated to the timing of movements? How stable are these representations over days and weeks? In what ways do these representations change with learning? These are fundamental questions to which we still, after many decades of research, lack clear answers to.

*Previous work in non-human primates and songbirds*

Two model organisms have primarily been used to study the neural mechanisms underlying the learning and production of movement sequences – non-human primates (mainly rhesus monkeys) and songbirds[26]. Non-human primates have the distinct advantage of being evolutionarily very close to humans and hence the functional organization of their brains is largely similar to ours. Furthermore, non-human primates are thought to be capable of learning tasks more complex than most other model organisms. However, ethical, practical and financial burdens[17] make primates ill-suited for systematic large-scale lesion studies. Hence, these previous studies mostly correlate single unit activity in different brain areas to ongoing behavior and attempt to infer their roles from these correlations. Despite decades

of work using this experimental approach no clear consensus has emerged - even the functional role of the primary motor cortex in primates is still debated[88]. Todorov summarized this confusion by citing studies that correlate single unit activity in monkey M1 to a variety of movement features: “M1 firing was also correlated with arm position, acceleration, movement preparation, target position, distance to target, overall trajectory, muscle coactivation, serial order, visual target position and joint configuration”[52]. One report mentioned that “… all types of neuron that were looked for were found, in nearly equal numbers”[89]. Similar to the lack of consensus on the functional role of M1, not much is known about the role of other central motor areas, especially as it pertains to the neural control of movement sequences. Various correlational studies have implicated M1, SMA, pre-SMA and PFC in largely overlapping roles[14,46,90–94].

The songs of zebra finches have recently emerged as an alternative model system for the studying complex learned motor sequences[87,95–97]. This has proved much more tractable and has resulted in a relatively strong consensus on the role that different brain areas play in the learning and production of birdsongs. Two major neural pathways are thought to be involved in song production and learning: the motor pathway and the anterior forebrain pathway. While the motor pathway is important for the production of learnt song, the ‘anterior forebrain pathway’ (AFP), which is homologous to basal ganglia thalamo-cortical loops in mammals, is thought to be important in song learning.

However, the songbird is limited in many respects as a model for human motor sequence learning. One fundamental drawback is that songbirds have specialized neural circuits dedicated to song production and hence neural mechanisms of song production and learning may not generalize to other kinds of motor sequences[26]. More generally, the homology between bird brains, with their notable lack of a cerebral cortex, and mammalian brains is tenuous. Finally, birdsong is also behaviorally less flexible than the motor sequences of primates, like sequences of arm movements, in that zebra finches sing only one song, which has a strong innate component[98].

*A new motor skill learning paradigm in rats*

These considerations combined with the drawbacks of primates have motivated the use of rodents as model systems for studying motor sequences. Not only do they have many of the same brain structures as primates but we are now able to train them to perform behaviors similar in complexity to that of primates using the high-throughput training system described in Chapter 1 (see Figures 1.3). To establish rodents as a model system for complex motor skill learning, Kawai et al[99] developed a new motor skill learning paradigm that is similar to the song of zebra finches in that they comprise a sequence of highly stereotyped and temporally precise movements.

In this task, animals were rewarded for pressing a lever twice with a precise interval between the two presses. After several weeks of gradually narrowing the range of rewarded inter-press intervals, each animal developed highly complex and extremely stereotyped but idiosyncratic sequence of movements of their limbs, necks and the whole body to generate the target inter-press interval. They then investigated the role of the motor cortex in this behavior and, contrary to expectations, found that bilateral lesions to the motor cortex did not impair execution of the learned behavior. Post-lesion, animals still performed the task at similar levels of precision and with the same sequence of stereotyped movements strongly indicating that motor cortex was not the site that generated the pattern of neural activity necessary to produce the behavior. However, when motor cortex was bilaterally lesioned in a naïve animal prior to any training, none of the animals (n=11) were able to learn the task to criterion. In particular, Kawai et al found that the precision of motor output in these animals was substantially and qualitatively lower than that of controls. Animals were particularly deficient in their ability to learn to withhold unwanted lever presses. The task required animals to wait 1.2s after an unrewarded trial before initiating a new trial. All control animals learnt this aspect of the task while all but one lesioned animal failed to learn this.

These lesion studies suggest that the role of the motor cortex changes over the course of learning a motor sequence. We recorded neural activity from single units in the primary motor cortex of rats using

the continuous long-term recording setup described in Chapter 2 in order to understand how neural activity changes with learning.

## Results

### *Experimental design*

Here, we report data from recordings in 4 rats learning the precise sequential lever press task developed by Kawai et al during various phases of training. While the goal was to record continuously for weeks as a naïve animal became fully proficient after weeks of training by combining the automated training and the continuous long-term recording systems described in the previous chapters, due to declining recording quality over time, this proved not to be feasible. Therefore, we recorded neural activity in the early stages of learning from 2 animals as soon as they were first exposed to the task and from late stages in 2 animals that had already become proficient in it.

After training 2 naïve animals to associate a lever press with a reward tone followed by a water reward, an array of 16 tetrodes was surgically implanted in the output layer of the motor cortex (see Methods for details). After a 1 week recovery from surgery, animals were put back in their home-cages and were gradually shaped using the automated training system of Chapter 1 to press the lever twice in rapid succession with a target inter-press interval of 700ms. Recordings were performed continuously during this period in a completely hands off manner using the system of Chapter 2. A similar procedure was used to record from the motor cortex of 2 other rats with the distinction that the electrode array was only implanted after the behavior was already well learnt. For these rats the array was implanted on the side contralateral to the paw used for the first lever press in the sequence.

### *Behavioral performance*

Animals learning this task have highly variable behavior before learning and converge to performing very precise movements after[99]. As expected, when the precision of behavior was measured simply by the distribution of inter-press intervals, we found that they were very wide in the two rats in which data was recorded early in training as compared to the two rats that had already learned the task

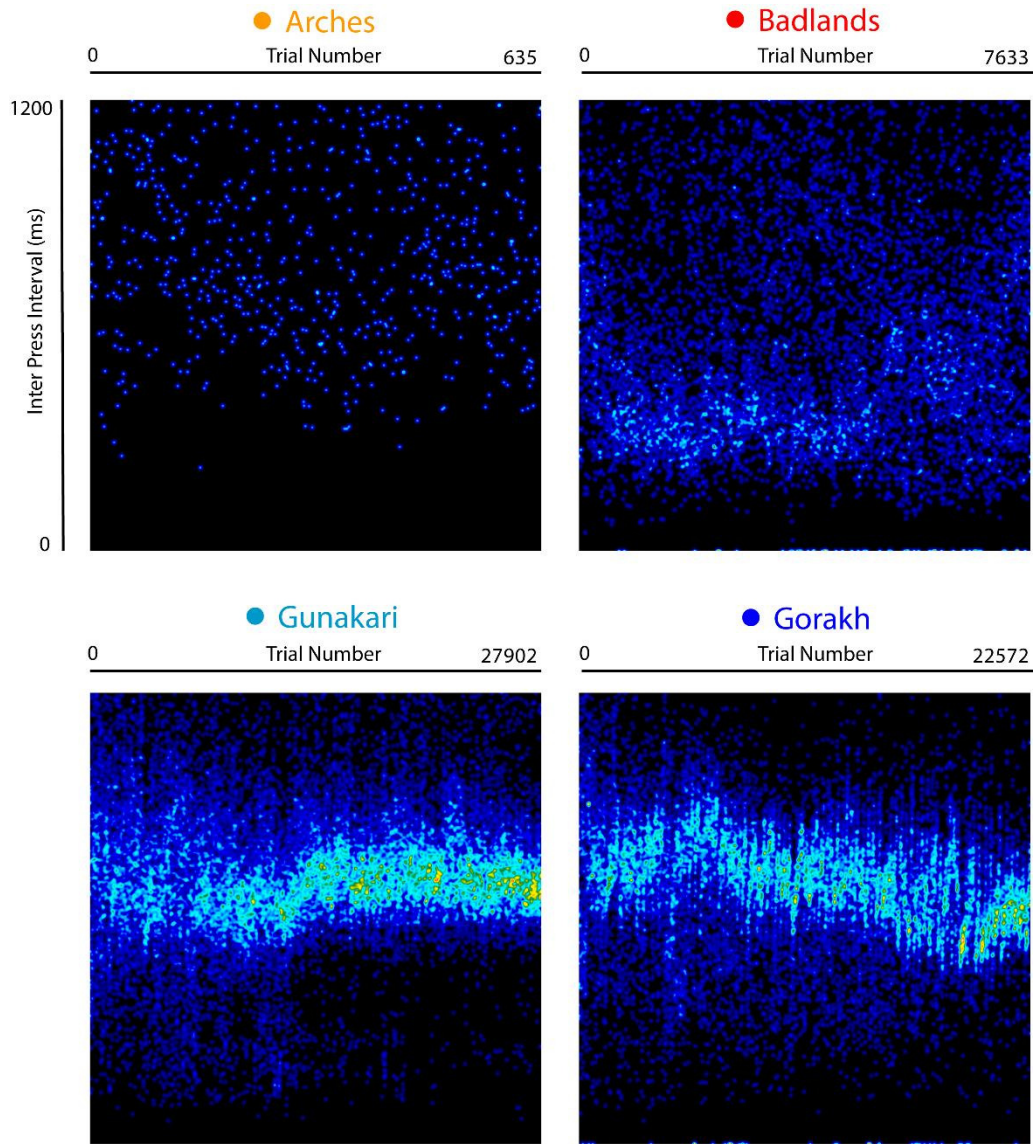



**Figure 3.1:** Distribution of inter-press intervals for the duration of the recordings. Animals were trained to press a lever twice in rapid succession with a target inter-press interval of 700ms. Rats 'Arches' and 'Badlands' comprise the pre-learning group and rats 'Gunakari' and 'Gorakh' the post-learning group. The heatmaps show the distribution of inter-press intervals with black corresponding to no presses and white to the most frequent presses. The color associated with each rat is reused in Figure 3.4. All trials, rewarded and unrewarded, comprising at least two lever presses are included.

(Figure 3.1). We recorded during the first week of training from rat 'Arches' during which it performed a total of 635 trials with two-lever presses each. No unit was present in any of the tetrodes after a week and hence the recordings were stopped. Another rat, 'Badlands' performed a total of 7633 trials in its first two weeks of trainings after which the recording quality was again too poor to continue. Since recordings from both these animals did not persist for the duration necessary to become proficient at the task, we implanted tetrode arrays in the motor cortex of two other rats after learning the task. These rats, 'Gunakari' (27902 trials) and 'Gorakh' (22572 trials), were recorded for 5 weeks as they performed the task at a high precision in a stereotyped manner.

*Diversity of neural representation*

We used the spike sorting pipeline described in the previous chapter to extract spike times from identifiable single units in all 4 animals. Units had to be clearly isolated from other units and noise for the full duration of a training session to be included in this analysis. Only sessions with at least 10 rewarded trials were included in the analysis. Furthermore, if a unit fired no spike in more than 5 trials in a session, then the unit was excluded from the analysis. Using this criteria, we were able to isolate 115 single units from Arches, 94 from Badlands, 76 from Gorakh and 39 from Gunakari for a total of 324 units during the hour long training sessions. This number overestimates the total number of distinct neurons since the same unit can be counted multiple times if its waveform become inseparable from noise and hence cannot be reliably tracked in between training sessions. If units with similar spike waveforms and similar functional properties recorded on the same tetrode at different times were to be counted as one unit, we estimate the total number of distinct units to be as little as half of 324. Many of these units were recorded for multiple training sessions. Across the 4 animals, we recorded 132 units for at least 2 training sessions and several units were recorded for over a week.

Even very early on in learning, the neural representation of a lever press in the motor cortex is dependent on its sequential context for many task relevant neurons. Figure 3.2 shows example activity

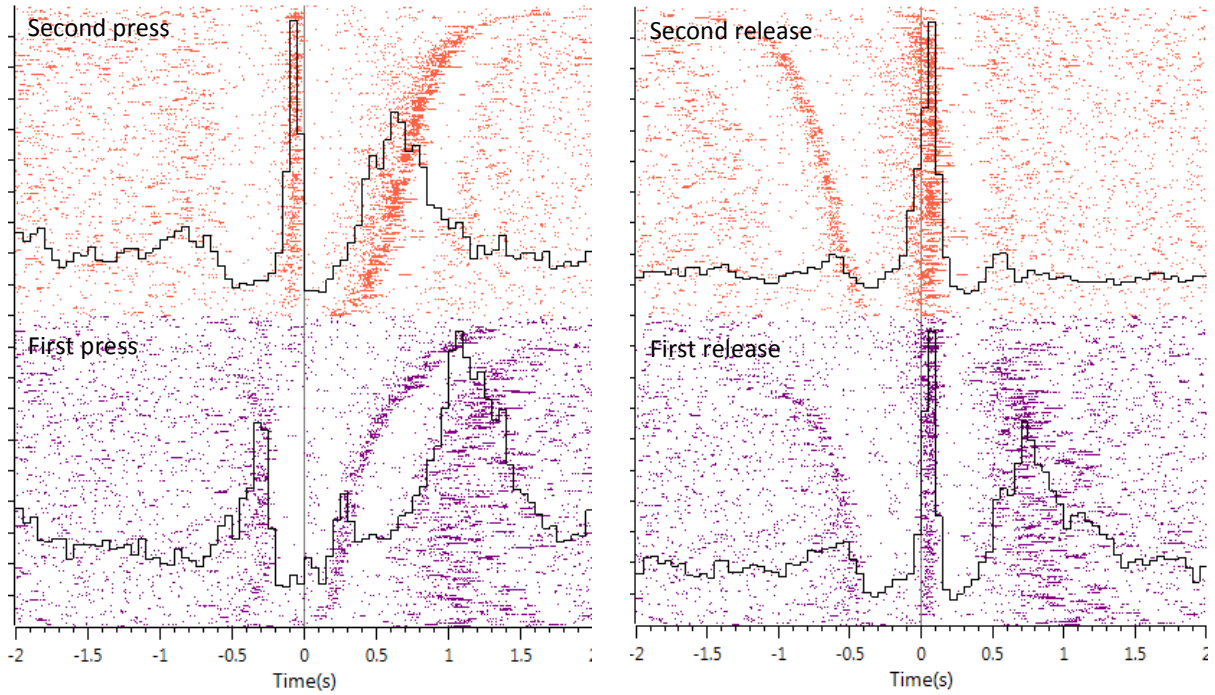



**Figure 3.2:** Spike rasters from an example task related neuron. This figures shows a 2x2 panel of spike rasters. Each panel shows the response of the neuron during 200 rewarded trials in a session. Each trial corresponds to a row in the panel and each tick corresponds to a spike. The x-axis of each panel encodes time relative to a behavioral event during the trial. The four events that define time 0 in the four panels are the times at which the lever was pressed (left panels) and released (right panels) for both the first (bottom panels) and the second (top panels) lever press of the trial. The solid black lines are the peri-event time histograms (PETHs). The height of the black lines at each 100 ms time bin is proportional to the average number of spikes fired by the neuron during that time bin.

from a neuron in rat 'Badlands' that is silent in the 200ms preceding the first lever press but fires a burst of spikes in the 100ms preceding the second press. This figure is a 2x2 layout of spike rasters. Each panel shows spike times of the same unit aligned to either the moment of engaging the lever (left panels) or releasing it (right panels). The bottom panels are aligned to the first lever press in a trial and the top panels are aligned to the second one. Each row in each panel corresponds to a single trial and each tick denotes the time at which the neuron fired. The rows in a panel are sorted by the duration of the lever press, i.e. the time between releasing and engaging the lever, with the bottom rows corresponding to the fastest presses and the top rows the slowest. The solid black lines in each spike raster correspond to the peri-event time histograms (PETHs, i.e. the average firing rate across trials).

Note that this neuron has a non-zero baseline firing rate which starts increasing about 700ms before the lever is first depressed and continues increasing for several hundred milliseconds (bottom left panel). Then the neuron abruptly stops firing until the lever is released at which point it proceeds to fire a burst of action potentials for 100ms (bottom right panel). The neuron becomes silent as soon as the lever is depressed again (top-left panel) and fires a second burst of action potentials when the lever is released a second time (top-right panel). Also, unlike the projection neurons of the HVC in zebra finches this neuron represents the behavior in a non-sparse manner by modulating its firing rate at multiple times during a motor sequence.

While a set a spike rasters aligned to various time points in the motor sequence as in Figure 3.2 forms a relatively complete characterization of the functional properties of the neuron and works well for visually examining how neural activity correlates with behavior, a systematic analysis of the full set of recorded units requires a more compact representation. The trial-by-trial variability in the temporal dynamics of the behavior, which takes the form of large variations in the duration of lever presses in this animal, makes average firing rates computed at times far from the point of alignment not very meaningful as they correspond to very different limb configurations and movements even just a few hundred milliseconds away from the point of alignment.

To systematically characterize the diversity of neural activity in the motor cortex of rats engaged in this task, we computed peri-event time histograms for each of the 324 units by separately aligning the spike times from each unit to 5 behavioral events – time of depression & release of the lever for each lever press and the time of reward delivery. PETHs were defined as the vector of average spike counts in each 100ms bin aligned to these events. One second of data centered on each lever press event (10 100ms bins for each of the 4 lever press events) and two seconds of data centered on the time of reward delivery (20 100ms bins) was used to compute PETHs. Therefore, each unit was characterized by a 60-dimensional vector, the spike count in each one of the 60 bins. Note that since some events were separated by less

than 1 second in some trials the same spike could be counted twice – for instance the bin corresponding

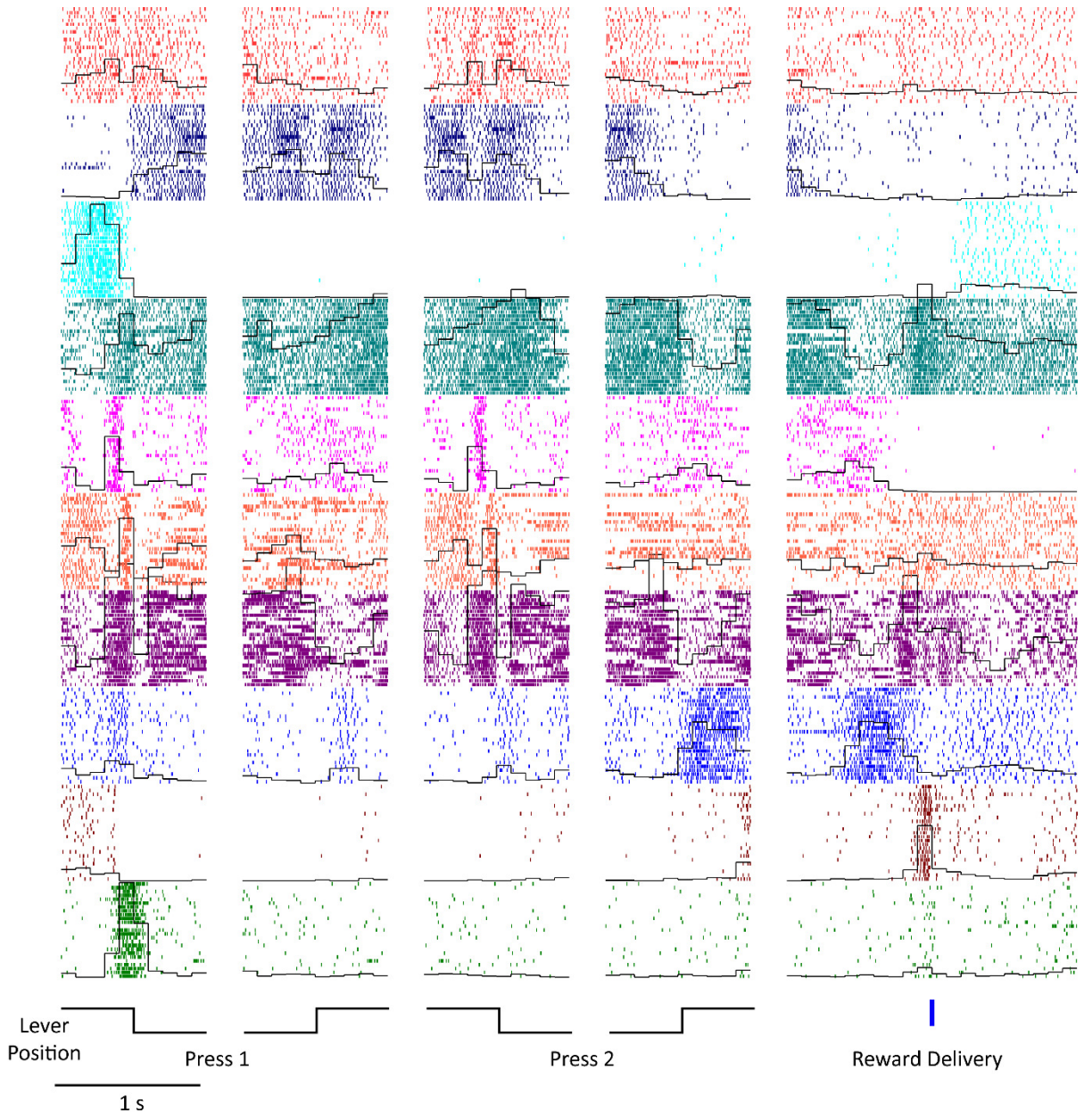



**Figure 3.3:** Example spike rasters and PETHs of 10 neurons. The spike rasters are similar to the ones shown in Figure 3.2. All the spike ticks from a neuron are coded with the same color and data from 25 rewarded trials are shown for each neuron. The five columns show spike times relative to five behavior events, 4 lever press events and 1 reward delivery event. The schematic below the rasters show the time of these events. The first column corresponds to spike times relative to the first lever press of a trial and second column corresponds to spike times relative to the first lever release. Similarly the third and fourth columns show spike times relative to the second lever press and release and finally the fifth column shows spike times relative to the moment of reward delivery. As before the solid black lines correspond to PETHs.

to 0.4s-0.5s after engaging the lever might be the same as the bin corresponding to 0.4s-0.5s before releasing the lever.

Figure 3.3 shows spike rasters (25 trials each) and PETHs from 10 example units aligned to the 5 behavioral events to illustrate the diversity of neuron types in the motor cortex of rats. The first (bottom-most) example unit fires a burst of spikes for about 200ms at the time of the first lever press but is largely silent at other times. The second unit fires a burst of spikes during reward delivery and has a below-baseline firing rate when the lever is pressed. The neural activity of the sixth unit is largely invariant to the sequential context of the lever press (unlike the example unit shown in Figure 3.2) in that the PETHs aligned to the first and second lever presses are very similar to each other. We found neurons that fire a single narrow burst of spikes at a particular phase in the task, neurons that fire a temporally broader burst of spikes, neurons that are tonically active, but modulate their firing rate at various phases of the task, and neurons that are strongly suppressed during certain phases of the task.

*Comparison of neural representation pre and post-learning*

Despite the fact that motor cortex is not necessary for the behavior after it is learnt, single unit activity in the motor cortex of rats is exquisitely correlated to behavior both early in learning and after animals are proficient. To find the fraction of task-related neurons, we calculated whether neural activity in any of the 60 time bins was significantly modulated. We defined significant modulation as a deviation from the average firing rate large enough to occur less than 1 in 100,000 times by chance. With this definition of modulation, a neuron was classified as task-related if it was significantly modulated during at least one of the 60 time bins. 100% of neurons recorded after learning were task-related despite motor cortex not being necessary for the task. At least 85% of the neurons recorded pre-learning were also task-related. Since many sessions early in learning did not have enough trials to detect small modulations, the number is likely to be an underestimate.

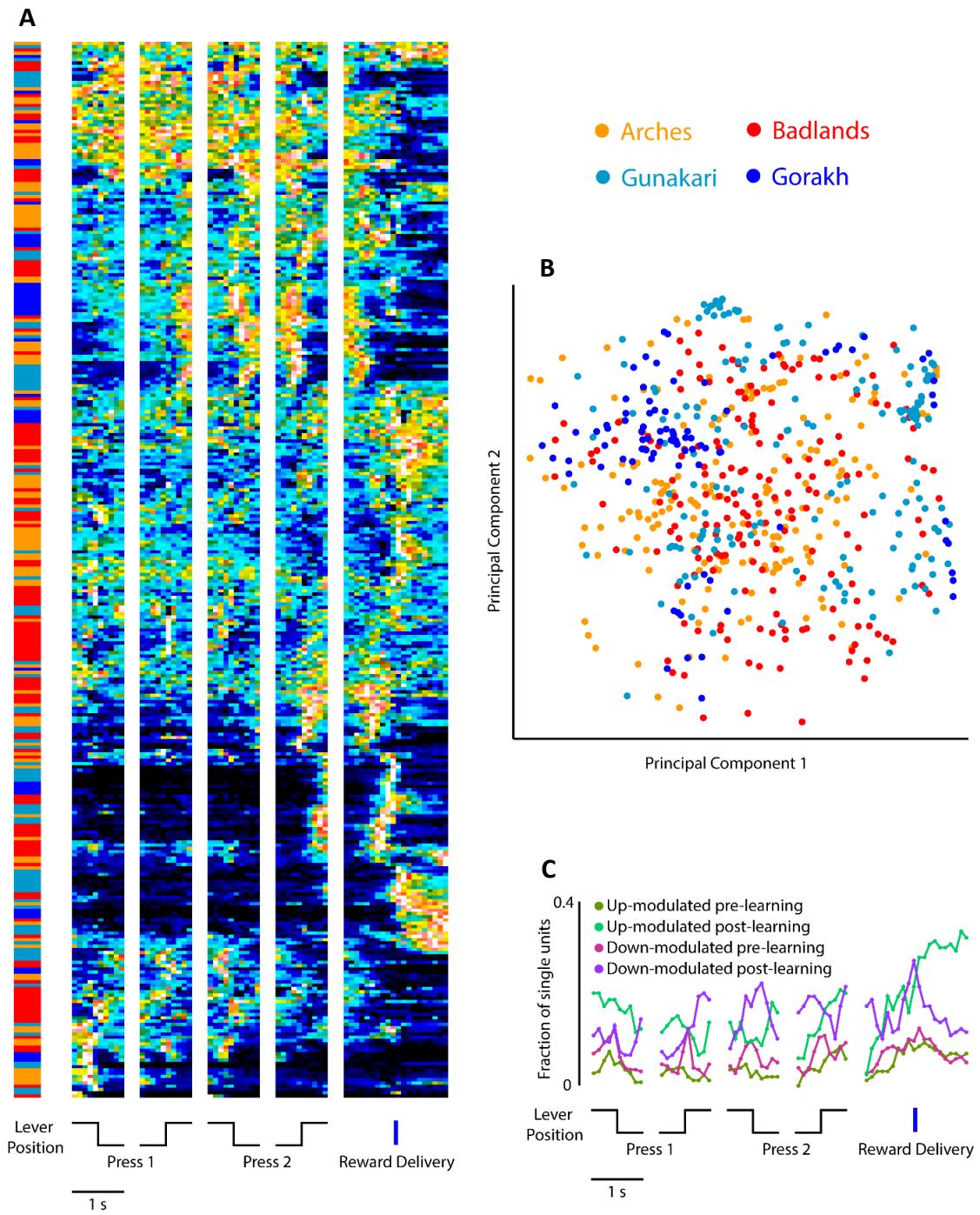



**Figure 3.4:** Diversity of neural representation. **A.** PETHs of all 324 neurons sorted by similarity. Each row shows the PETH of a neuron (calculated from the spike rasters as in Figures 3.3) using a color map (black - no firing, white - maximal firing). The colored bar on the left shows which neuron belongs to which rat. **B.** Two-dimensional principal-component projection of the 60-dimensional PETH vectors of all 712 neuron-session pairs color coded by animal. Points from the pre-learning group (reds) and the post-learning group (blues) are intermixed indicating that neurons of all types were present in both stages of learning. **C.** Fraction of single units that were modulated during each phase of the task. Data is broken down by learning stage (pre vs post) and direction of modulation (increase vs decrease).

Furthermore, neurons were correlated to the behavior in very diverse ways and, to first order, there was no difference in how neurons represented the behavior before and after learning (Figure 3.4). To assess whether certain neuron types were only present pre or post-learning we visualized the PETHs of all 324 units separately normalized to the maximum and minimum firing rate of each unit (Figure 3.4A). Each row in the figure shows the PETH of a single unit and the rows are ordered by similarity computed using hierarchical agglomerative clustering (see Methods). Figure 3.4A also shows which rat each unit comes from with the same color code as Figure 3.1. Note that for virtually every type of activity pattern units from both pre- and post-learning groups are represented. Also, the similarity based ordering shows the full set of 324 units subdivides into subsets of units with similar patterns of neural activity suggesting that neurons in the motor cortex could potentially be classified into a smaller number of functional types. To assess whether different phases of the task were likely to be differentially represented in the neural activity of single units, we calculated the fraction of units that were modulated during each of the 60 time bins (Figure 3.4C). While at least some neurons are modulated during every phase of the task, neural activity seems to be particularly strongly modulated by reward delivery. To visualize the space of activity patterns spanned by the motor cortex neurons, we plotted a low dimensional projection (first two principal components, see Methods for details) of the 60-dimensional points characterizing PETHs of each unit (Figure 3.4B). For each session that each neuron was recorded for, we plotted a point color coded by the animal the neuron was recorded from. Since the points corresponding to pre and post-learning groups are intermixed and span the full space, this supports our contention that motor cortex neurons encode this behavior in similar ways before and after learning.

*Temporal precision of neural representation*

The activity of some units was very precisely aligned to one (or more) of the 5 events. To quantify the level of temporal precision in the neural activity of single units we constructed a spike count classifier to separate adjacent time bins (see Methods for details). If the activity of a neuron can be used to separate

one 100ms time bin from the next (for instance the time bin just before a lever press to just after) but not adjacent 50ms time bins then the neural activity can be said to be precise to between 50ms-100ms. We defined the temporal precision of each unit as the smallest bin size that still allowed at least one pair of adjacent time bins to be clearly separated. The criterion for clear separation was a classification error rate of less than 20% which corresponds to a p-value of less than 1 in 1000. Again there was very little difference between pre and post-learning groups. Among the units recorded in rats post learning, 6 units were temporally precise to between 16ms-32ms, 18 units to between 32ms-64ms, and 15 units to between 64ms-128ms. The corresponding numbers for early in learning was 4, 19 and 23. No unit in either group was precise to sub 10ms resolution at least with respect to the 5 events that were considered. It's possible that identifying events from more detailed kinematic data like forepaw position would uncover even more precise alignment of the neural activity to behavior.

*Stability of neural representation*

While the patterns of activity of the set of single units that we recorded was very diverse, each unit stably maintained its functional properties across training sessions. Between the 4 animals, we were able to isolate and continuously track 132 units, each spanning at least 2 training sessions separated by 5 hours. Several neurons were recorded continuously for over a week. In every single instance, the PETH of the same unit in different training sessions was very similar. Figure 3.5 shows spike rasters and PETHs from 3 training sessions each for 3 sample neurons. The average Euclidean distance between the 60-dimensional PETH vectors belonging to the same cluster (i.e. the same neuron during different sessions) was significantly smaller ($p < 0.05$) than the average distance between vectors corresponding to different neurons for every single neuron.

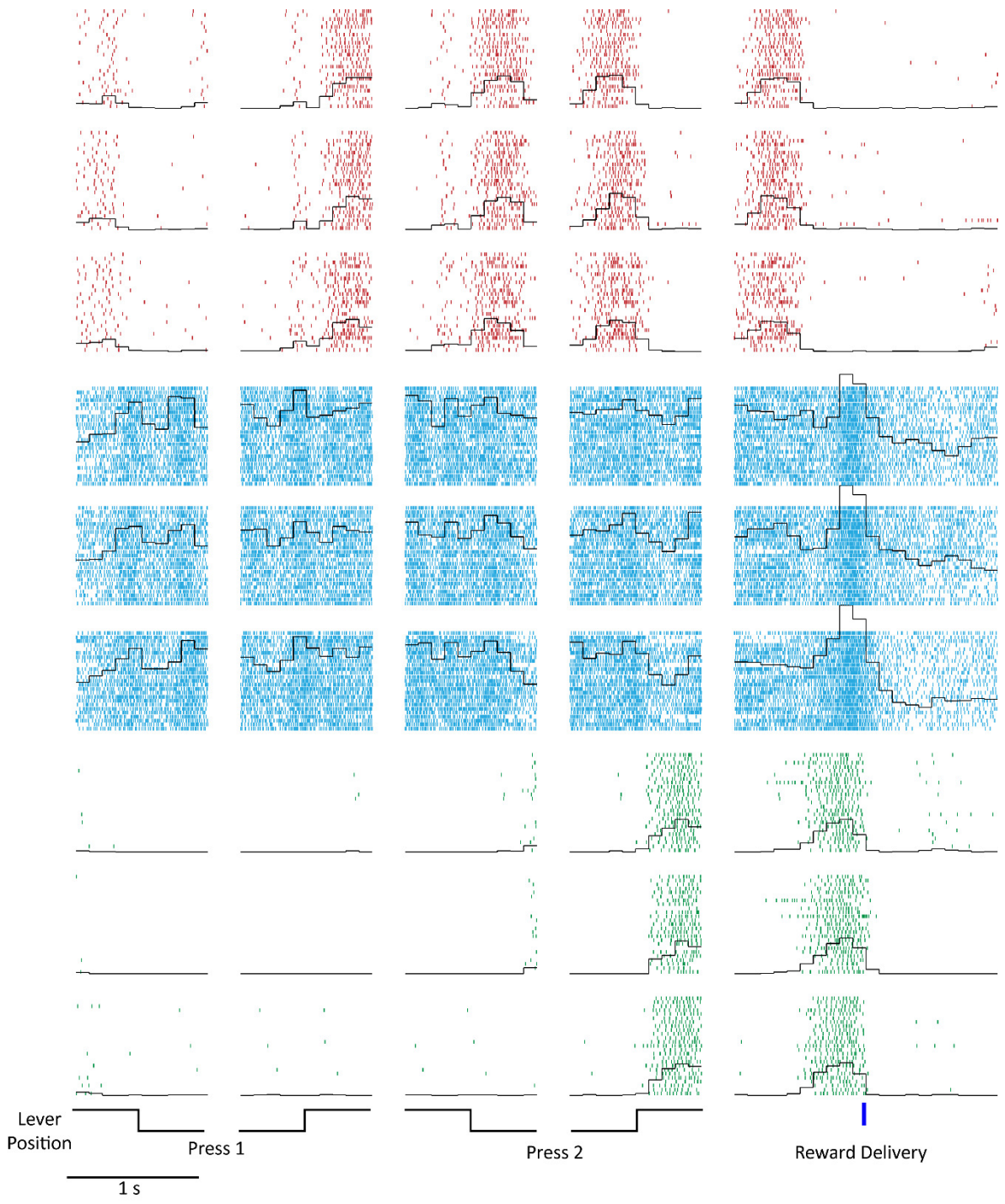



**Figure 3.5:** Similar to Figures 3.3. Here data is shown from 3 separate sessions for each of 3 neurons. The pattern of firing of each neuron in response to behavioral events remains very stable over sessions.

In a fifth rat, we were able to track a single neuron in the motor cortex continuously for a period

of 41 days (from the 18th to the 59th day since the start of training) as the behavior became more precise. This is the same neuron as the one show in Figure 2.15 of the previous chapter. While the gross structure of the PETH of the neuron remained constant - it fired a burst of spikes just after the first lever press - the firing rate during the burst decreased steadily over time (Figure 3.6). Firing rate during the burst decreased from a peak of 133.5 Hz at the beginning to 65.5Hz at the end.

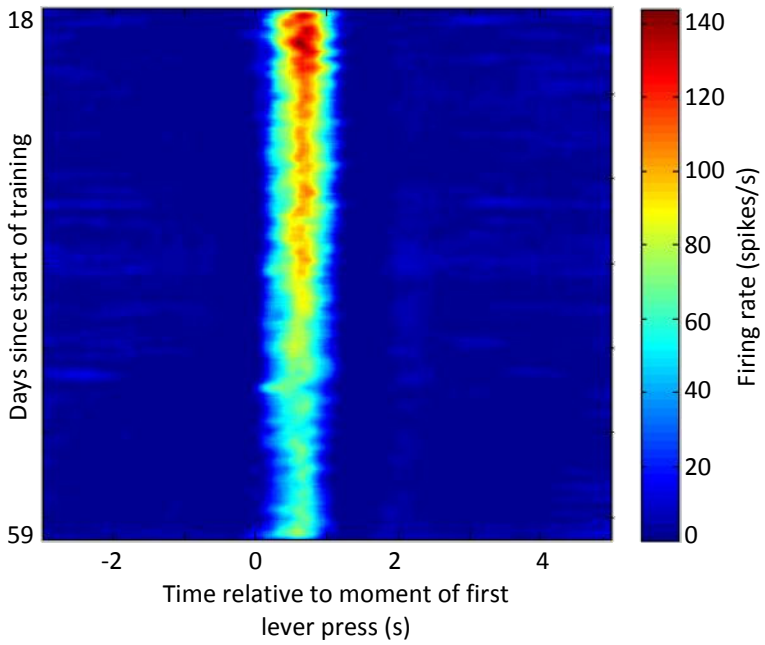



**Figure 3.6:** PETH of a neuron continuously recorded for 41 days. Each row in the density plot above corresponds to the PETH from one session computed from all the rewarded trials in the session. The animal had 2 sessions per day. This neuron fires a burst of action potentials just after the first lever press of a trial (time 0) and a second weaker burst about 2 seconds after the first tap.

## Discussion

Inspired by the birdsong model system[87,95], we trained rats to perform a sequence of highly stereotyped and temporally precise movements by requiring them to press a lever twice within a narrow range of inter-press intervals (rats 'Gunakari' and 'Gorakh' in Figure 3.1). Then we recorded a population of single units from the motor cortex of these rats as they performed the movements over and over again. We found that every neuron we recorded in rats 'Gunakari' and 'Gorakh' was significantly modulated

during at least one phase of the task. Many neurons fired bursts of action potentials precisely aligned to the first, second or both lever presses (Figures 3.3 - 3.6).

That neurons in the motor cortex represent ongoing movement of the limbs and body in diverse ways is not surprising given existing reports in both the primate[52,92] and the rodent literature[58,59,100–103]. However, after a 5-day recovery from bilateral lesions of the motor cortex task performance was not impaired[99], suggesting that neural activity in the motor cortex is not necessary for generating the muscle dynamics that lead to the behavior. Why then might the activity of these neurons be so exquisitely correlated to ongoing behavior? What if any role does motor cortex play in the context of producing a complex sequence of motor acts?

We suspect that the phenomenon of brain areas not needed to produce a given behavior nonetheless having neural activity correlated to the behavior is very widespread. Due to the relative paucity of lesion studies in non-human primates, this discrepancy is rarely observed. Since generating action potentials is thought to account for a large fraction of the energy consumption of brains, producing 'unnecessary' spikes is energetically wasteful. Therefore, if we wish to retain the model that neural circuits operate in an energetically constrained regime then this suggests that motor cortex must be playing some, as yet unidentified, role in the production of the sequential lever pressing behavior.

One plausible hypothesis is that sub-cortical structures generate the neural dynamics that cause this behavior[104,105] and pass along an efference copy of their activity to the motor cortex. The motor cortex might then use the efference copy to monitor the behavior and if needed direct sub-cortical structures to alter the motor programs to react to changing conditions thereby making the behavior more robust. For instance, if the reward contingencies change such that 500ms inter-press intervals are rewarded instead of 700ms ones, then the motor program needs to be altered appropriately. However, bilateral motor cortex lesions do not impair the ability of animals to adapt[99] to this change making this hypothesis less tenable. Perhaps other changes that require a more radical alteration of the motor

program like requiring greater force to depress the lever or placing an obstacle near the lever that interferes with the learned motor program might uncover the necessity of the motor cortex. Another possibility is that these neurons are simply reporting somatosensory and proprioceptive information since neurons in the motor cortex of rodents are known to respond to touch[106].

While motor cortex is not necessary to perform the task, it is needed to learn it[99]. Therefore, we were surprised to find that at a gross level, neural activity in the motor cortex was remarkably similar between the animals that had not yet learnt the task and the animals that had already mastered it (Figure 3.4). This finding strongly cautions against using correlative evidence from neural recordings to infer 'roles' for different brain areas. Lesion studies show a clear dissociation in the involvement of the motor cortex in learning vs executing the task, but single unit activity is nonetheless quite similar. We haven't yet ruled out the possibility that subtle features of population neural activity like pairwise correlations between neurons does indeed change with learning. This analysis would be greatly aided by recording from the same population of neurons throughout learning allowing within subject rather than across subject comparisons. While we were unable to do so because the quality of our recordings degraded substantially after a week, other researches have been able to successfully maintain recording quality for longer durations[53–55].

Nonetheless, we were able to track many neurons for at least several days and in one case for over 40 days. We found that the pattern in which a neuron fired during each trial remained very stable at least at a gross level (Figures 3.5 and 3.6). Over time, a neuron that fired a short burst of action potentials just after the first lever press did not switch to becoming a neuron that was tonically active throughout the trial. Neurons maintained stable functional identities during the phase that the task was actively being learnt and after the behavior was consolidated into a stereotyped sequence. This is consistent with previous studies that found a high level of stability in the firing rate and directional tuning of neurons in the primate motor cortex during a center-out task[107,108]. In contrast several other studies found

unstable single-neuron representations but stable population representations with stability increasing with learning and practice[58,59,109,110]. However, these studies did not rule out the possibility that the variability in single-neuron representation was due to subtle underlying movement variability.

The decrease in firing rate over time of the neuron in Figure 3.6, if found consistently for many neurons, might suggest a diminishing role for the motor cortex once a behavior is consolidated. However, this would require ascertaining that the muscle dynamics remain stable over this period. While we can use video tracking of the forepaw to measure its position and ensure that the movement sequence remained unchanged in the kinematic domain, we would need to measure EMGs for this argument to be fully convincing.

The automated training and continuous long-term training systems described in Chapters 1 and 2 now allows us to routinely collect large datasets like the one explored in this chapter. Leveraging the fact that we record neural activity continuously should provide numerous insights into previously unanswerable questions. What role does sleep play in the learning and consolidation of a motor skill? Can we find evidence of sleep replay in the motor cortex of rats? How do task relevant neurons respond in non-task contexts? If firing in a neuron is followed by firing in another neuron during the task, does this relationship hold outside of task context? If two neurons are highly correlated with each other during the task, does that remain true in non-task contexts as well? Can we find evidence of increased correlations between neurons or between a neuron and behavior after a bout of sleep? Technical improvements in the design of the electrode array to increase recording stability combined with further analysis should shed light on some of these questions.

## Methods

*Behavior*

Female Long Evans rats aged 10-12 weeks were trained to press a lever twice in rapid succession with a precise 700ms interval between the two presses using the automated training system described in Chapter 1. This is the same task as the one depicted in Figure 1.4A and described in the 'precise lever pressing task' subsection of the 'Behavioral training methods section' of Chapter 1. More details on the training protocol used can be found at [99].

*Implant and recording*

Standard tetrode construction techniques were used to make a set of 16 tetrodes from 10µm diameter nichrome wires. The set of tetrodes were assembled into a 4x4 grid with 0.5mm spacing between adjacent tetrodes. Each electrode was gold-plated using a mixture of gold-cyanide and carbon nanotubes[69] to an impedance as low as possible without shorting adjacent electrodes of a tetrode. This resulted in a target impedance of approximately 100kΩ. Standard aseptic surgical techniques were used to implant the array of tetrodes into the output layer of the motor cortex (on the side contralateral to the forepaw used for lever pressing) of each animal (centered at 2.5mm lateral to and 1mm anterior to bregma at a depth of 1.7mm). After a week of recovery from the surgery, animals were placed in their home-cages outfitted with the continuous tethering attachment described in Chapter 2. The recording hardware described in Chapter 2 was then used to continuously record from the implanted animals.

*Spike sorting*

The spike sorting pipeline described in Chapter 2 was used to first generate a conservative sorting of the putative spikes for all 16 tetrodes. Then the spike sorting viewer was used to manually inspect the quality of the sorting. Only well-isolated units with waveforms that that were clearly separate in shape from nearby clusters was used to compile the set of 324 units used for further analysis. Many units were tracked across multiple sessions resulting in a total of 712 unit-session pairs. Only sessions with at least 25 rewarded trials were used in all analysis. Because the choice of parameters in the spike-sorting

algorithm was very conservative, we rarely encountered errors where distinct units were merged into one. We encountered several instances of errors in the other direction where the same unit was split into multiple clusters either because the unit had a low firing rate and hence there were periods of inactivity or because the spike waveform of the unit changed by a lot in a short amount of time.

*Computing PETH and calculating significance of modulation*

For each behavior session that a neuron was recorded, average spike counts were computed for a set of 60 time bins - 10 100ms time bins centered on each of the 4 lever press events (time of pressing and releasing the lever for each of the two lever presses comprising a rewarded trial) and 20 100ms time bins centered on the time of reward delivery. The number of spikes in each bin during each trial was assembled into a $NX60$ matrix of spike counts from the set of spike times of the neuron where $N$ is the number of rewarded trials in the session. This matrix was averaged along its columns to produce a 60-dimensional vector of average spike counts – this is referred to as the PETH of the neuron.

To assess whether the average spike count in a bin was significantly different from the average firing rate of the neuron, indicating functional modulation, we used a random re-sampling approach. We randomly chose $N$ elements from the $NX60$ spike count matrix and averaged them to compute one sample of average spike counts from the null distribution. This process was repeated a 100,000 times to generate an estimate of the full null distribution. Let $\boldsymbol{x} = (x_1, \dots, x_{60})$ be the PETH vector and $\bar{x}$ be the average of all elements of $\boldsymbol{x}$. The p-value associated with each $x_i$ is then the fraction of values in the null distribution outside the range$(\ dx_i - \bar{x},\ dx_i + \bar{x})$, $dx_i = |\ x_i - \bar{x}|$. A stringent criteria of p-value less than 1 in 100,000 was used to decide whether the average spike count in any bin was considered significantly different from the full average partly to compensate of the 60 multiple comparisons being made for each neuron.

*Computation of the PETH colormap (Figure 3.4A)*

The minimum and maximum values of each 60-dimensional PETH vector was used to rescale each element of the vector to lie within the range $[0,1]$. The MATLAB 'linkage' command with the 'average' method was used to compute a hierarchical agglomerative clustering of the set of 324 60-dimensional vectors representing the PETHs of the full set of recorded neurons. When a neuron was recorded for more than one session, only the PETH from the first session was used. Then the 'optimalleaforder' command was used to construct an ordering of the 324 vectors with the resulting order having the property that elements near each other in the ordering were most similar. Then the matrix of PETH vectors was rearranged using this ordering and plotted to produce the heatmap shown in Figure 3.4A.

*Two dimensional principal component projection of the PETH vectors*

The full set of 712 rescaled PETH vectors for each neuron and each session was assembled into a 712x60 matrix and the mean along each one of the 60 dimensions subtracted. The singular value decomposition of this matrix was then used to project this set of 60-dimensional vectors along the first 2 principal components resulting a set to 712 2-dimensional points. Figure 3.4B show a scatterplot of these points.

*Calculation of temporal resolution*

If the firing rate of a unit dropped significantly 115ms after a lever press event with less than 10ms of trial-by-trial jitter, then the spike counts in the two 10ms wide time bins flanking this moment have very little overlap and the activity of the neuron can be said to be precise to less than 10ms resolution. Let $\{a_i\}_{i=1}^{N}$ be the set of spike counts of a neuron in all $N$ trials of session in a time bin aligned to one of the 5 behavioral events. Similarly, let $\{b_i\}_{i=1}^{N}$be the set of set of spike counts in the time bin following the previous one. A simple classifier can then be constructed to separate the pooled set of spike counts $\{a_i\} \cup \{b_i\}$ to minimize the number of errors in the classification. The classifier picks a threshold $x$ and assigns all values less than $x$ to one group the rest to the other. The threshold that minimizes the number of

misclassifications is then used. If there is little overlap in the range of spike counts spanned by each bin then the classifier performs extremely well. Conversely, if the spike counts in both bins span the same range of values then the classifier performs at chance level with 50% errors. We used a threshold of 20% misclassification rate to define 'easily separable'. By random resampling from our dataset, we found that this corresponds to a p-value of less than 1 in 1,000.

Let $T_a^b$ be the classification error of the spike counts in a pair of adjacent $b$ ms wide time bins centered at time $a$ relative to one of the 5 behavioral events. For a lever press event we first calculated the time when the neural activity was modulated in the maximally precise way, i.e. by calculating the minimum of $T_a^{256}$ for the range $-500 \leq a \leq +500$. We did the same across all 5 behavioral events using a range of $-1000 \leq a \leq +1000$ for alignment to the time of reward delivery. If at any time instant the classification error was less than 20% we considered the neural activity to be precise to less than 256ms resolution. We then repeated the same process for 128ms wide bins and again if classification error was less than 20% then for 64ms bins and so on. The smallest bin size with a temporal precision allowing less than 20% classification error was defined as the temporal precision of the neuron.

*Quantification of PETH similarity across sessions*

To confirm that the patterns of neural activity of the same neuron from different sessions were indeed very similar to each other, we computed the average Euclidean distance between the 60-dimensional PETH vectors associated with the neuron. Then we used random resampling from the full set of 712 PETH vectors to construct the null distribution and compute a p-value for each neuron.

## Attribution

Rajesh Poddar conceived and designed the experiments. Rajesh Poddar and Ashesh Dhawale performed the recordings. Rajesh Poddar analyzed the data and wrote the chapter.

# Supplementary File S1

## Supplementary Methods

*Training stages for the center-out task*

Rats were trained in three sequential stages, each consisting of multiple sub-stages. Progression required performing a pre-specified number of correct trials at a stage. The first stage acquainted animals with the behavior box, which included the following sub-stages (numbers refer to Figure 1.2):

- collecting water within 3s of it being dispensed from the reward spout ($1_0$);
- licking the reward spout causing water to be dispensed ($1_1$);
- collecting water within a specified time interval after a sound (1 kHz pure tone, 200 ms long) is played (sub-stages $1_2$, $1_3$, and $1_4$ associated with decreasing frequency of tone playback and time-intervals)
- touching the joystick to produce the reward tone and subsequent reward delivery ($1_5$).

The goal of the second stage was to get rats to press the joystick down by 2.5cm when the center LED was lit. Rats were gradually shaped to do this by increasing the required amplitude of movement over five sub-stages ($2_0 – 2_4$). Finally the rats were trained to press the joystick down when the center LED was on ($2_5$).

The third stage shaped rats to move the joystick left or right along the arms of the inverted Y, after moving it down, to perform the final task. At the beginning of the third stage ($3_0$) both cues (left and right) appeared with equal probability, there was no timeout for moving in the wrong direction and rats needed to move the joystick only 0.5cm in either direction. The cue frequencies, required amplitudes of movement, and the timeouts associated with incorrect joystick movements were gradually changed in 16 small steps ($3_0$ to $3_{15}$) until the rat was moving the joystick to its full extent in both directions.

*Training Stages for the precise lever pressing task*

Water deprived rats were initially trained to associate the water spout with a water reward by dispensing ‘free’ water. After this they were trained to associate a short 100ms pure tone with availability of a water

reward which was dispensed upon licking the water spout. Next, they were required to press the lever to trigger the reward tone and then to press the lever twice in succession for the same reward tone. Finally they were trained to wait 700ms between consecutive lever presses by gradually narrowing the reward window around this target interval. At the beginning of each training session, the rewarded inter-tap interval range was calculated based on the previous session's performance, such that animals received reward on ~ 35-40% of the trials.

**Supplementary Table 1**

| Component | Unit Cost | Quantity | Total Cost |
|---|---|---|---|
| Client Computers (Core i5-750, 4GB RAM) | $700 | 24 | $16,800 |
| NI - DAQ System - (NI-PCIe 6323, 2x RC68-68, 2x CB-68LP) | $800 | 24 | $19,200 |
| Server Computers (Core i7-950, 12GB RAM) | $2,000 | 2 | $4,000 |
| Custom behavior boxes (Plastic, LEDs, valves, etc.) | $500 | 48 | $24,000 |
| Common Infrastructure (Wire shelving, water supply system, audio-visual isolation boxes, etc.) | $3,000 | 1 | $3,000 |
| **Total** | **$1,395.83** | **48** | **$67,000** |

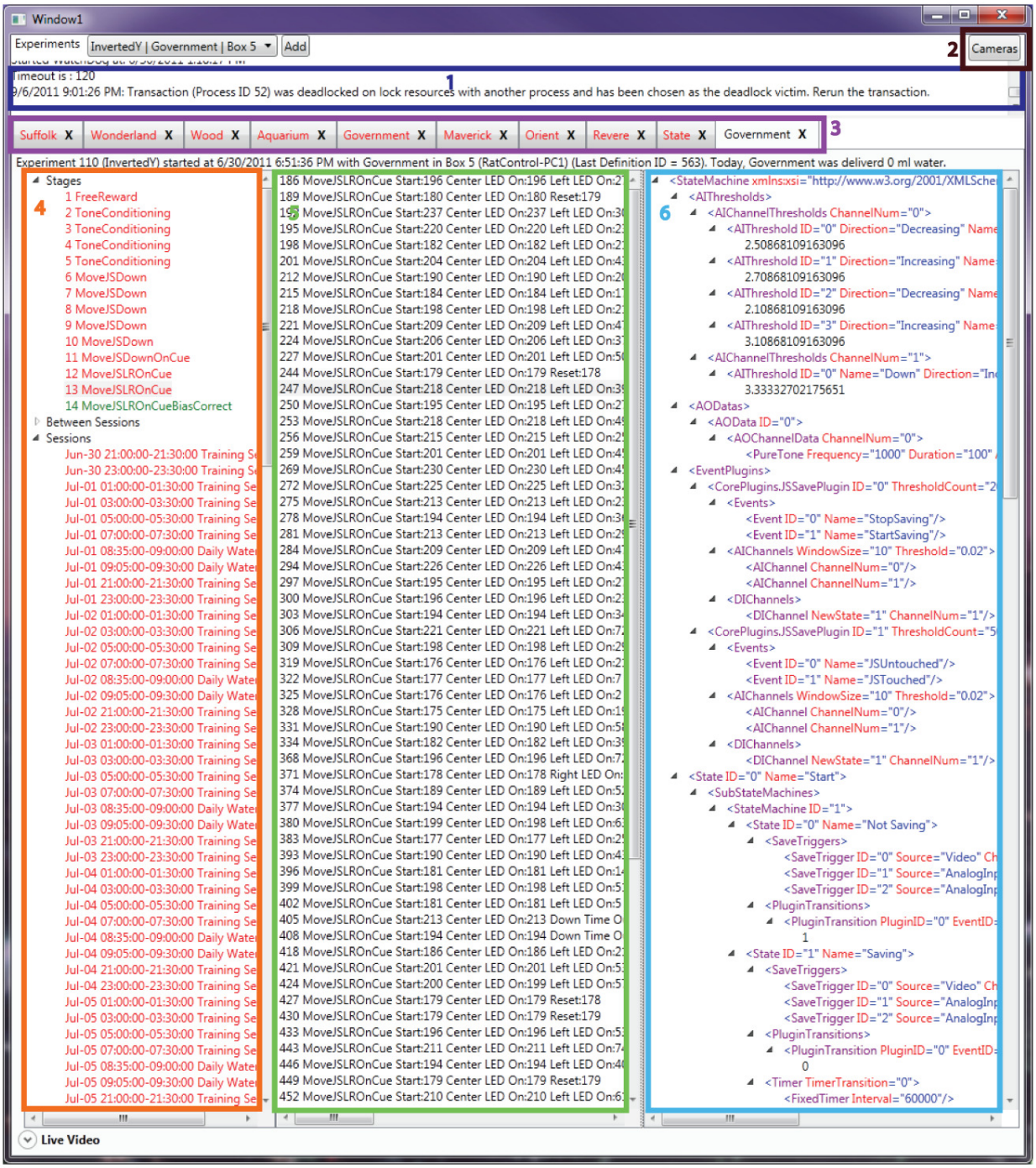



**Figure S1.1** A screenshot of *ARTSMonitor*, the behavior monitoring GUI of *ARTS*. This window shows an overview of behavior data for each rat being trained. 1. Messages from *ARTSWatchDog*, a program that continually monitors the entire system to make sure it is operating smoothly. 2. Allows the operator to view live streamed video of every rat. 3. Each tab in the window shows more detailed behavior data for each rat being trained. Critical pieces of information like how much water the animal received in the past 24 hours, what stage of training it is currently at are immediately visible in this interface. 4. Groups finite state machines by training stage or training session. Selecting an entry in this panel shows all the finite state machines that were executed within that stage or session. 5. Shows the number of entries to each state of a finite state machine, giving an immediate picture of the number and fraction of various trial outcomes over time. Double clicking a finite state machine in this panel opens a new window that shows detailed trial by trial behavior data as shown in Supplementary Fig 2. 6. Shows the XML document specifying the detailed structure of the finite state machine selected in the middle panel.

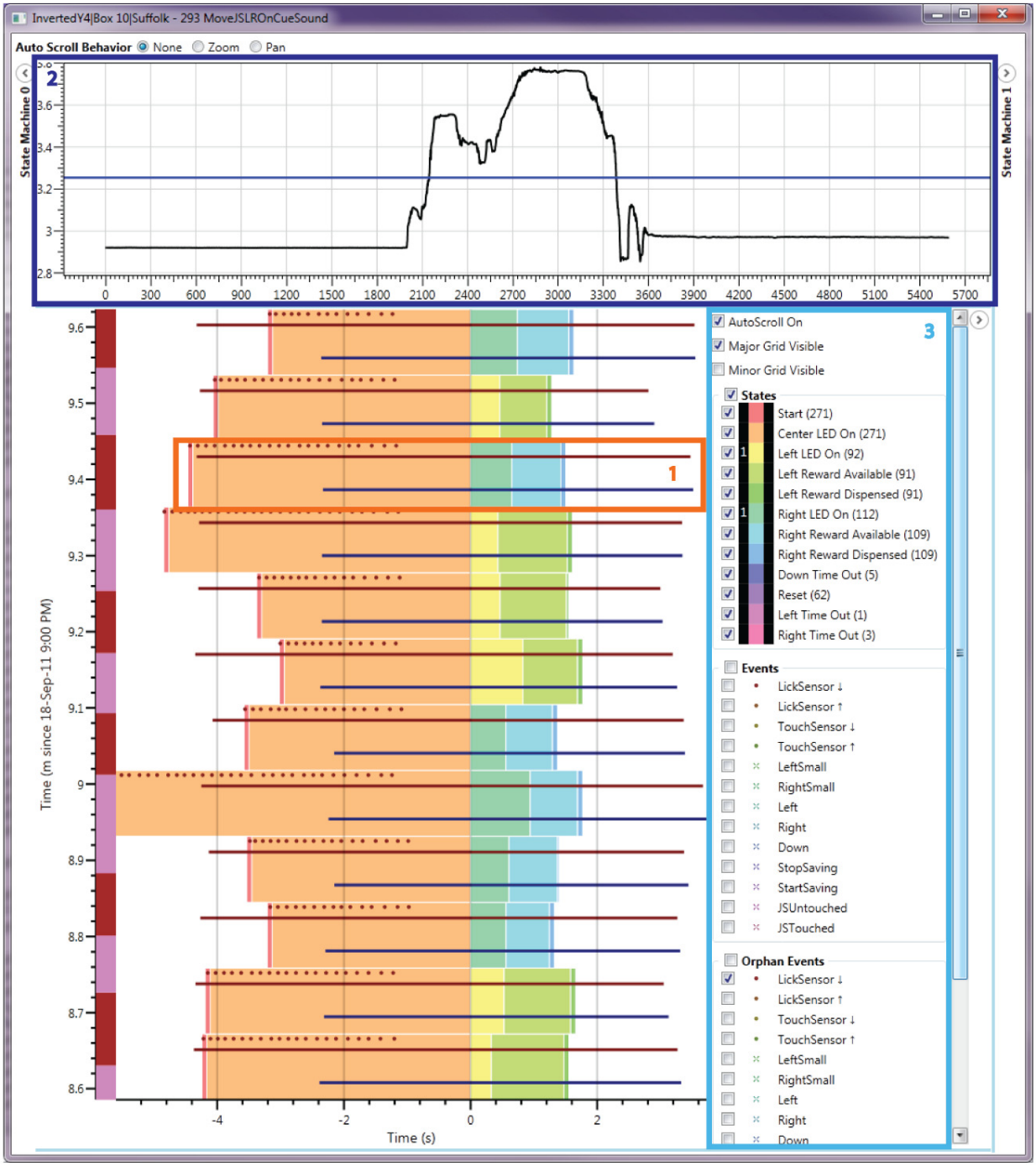



**Figure S1.2** A screenshot of the interactive data-visualization tool of *ARTS*. This window shows data from one finite state machine which can consist of several hundred trials over a 30-minute session. 1. Each row of the plot is a trial which comprises of a sequence of states each of which is identified by a color. Colored markers show the times of events (brown dots represent licking in this case) and colored lines show intervals for which some data was saved. Clicking on the brown lines allows viewing saved video from each trial. 2. Clicking on the blue lines shows the joystick trajectories on the top panel of the window. 3. This panel allows customization of the display by choosing the colors, marker styles and visibility of the various, state, intervals and events associated with the FSM being displayed. The display is highly interactive allowing interactive zooming, scrolling and alignment to onsets of chosen state entry events.

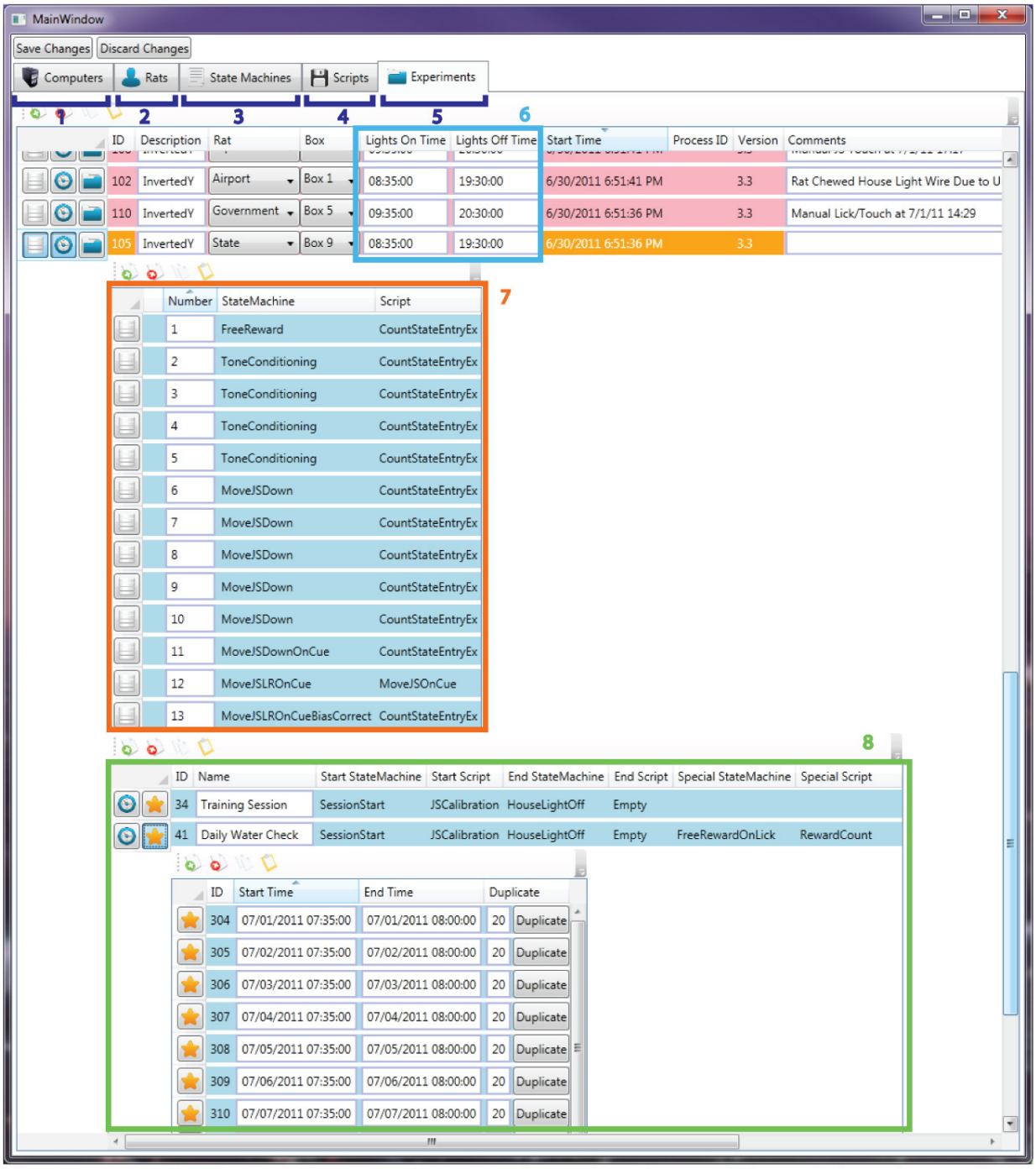
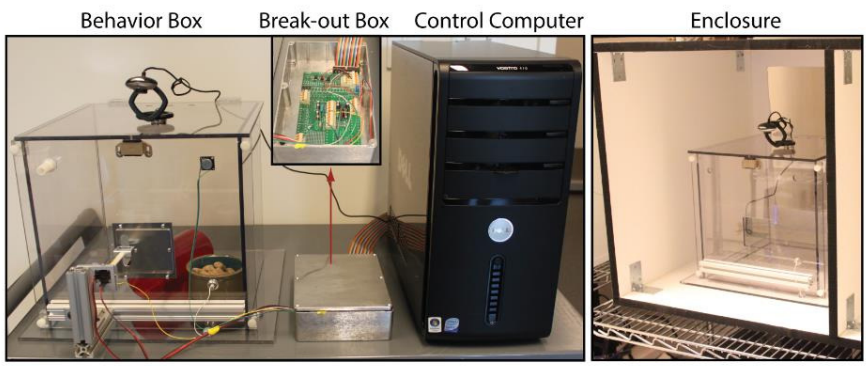



**Figure S1.3** A screenshot of *ARTSSpec*, the system configuration and protocol specification tool of *ARTS*, which provides a AGUI for interacting with the central *ARTS*database. The 'Computers' tab (1) allows configuring the various client computers comprising the *ARTS* system, and the behavior boxes connected to each computer. An XML document, the 'Box Definition', is associated with each behavior box and specifies the configuration of each box, like which actuators and sensors are part of the behavior box and which channels in the data acquisition card they connect to, making the system flexible enough to accommodate any behavioral paradigm. The 'Rats' tab (2) allows entering the names, weight, etc. of the animals being trained. The 'State Machines' tab (3) allows specifying the FSMs that correspond to each task automated by *ARTS* during training in the form of an XML document, which is a human readable and intuitive format for specifying Finite State Machines. *ARTS* contains XML schemas that specify the syntax of these documents allowing auto-completion and live syntax checking ('IntelliSense'), making entry of state machines user friendly. The 'Scripts' tab (4) allows specifying custom-written scripts that *ARTSTrainer* uses to evaluate behavioral performance and transition to subsequent stages if necessary. Finally the 'Experiments' tab (5), allows putting all this together, to specify the entire weeks-long training protocol which consists of the day/night light cycle (6), the FSMs and scripts associated with training stages (7), and the session structure (8).

A




B

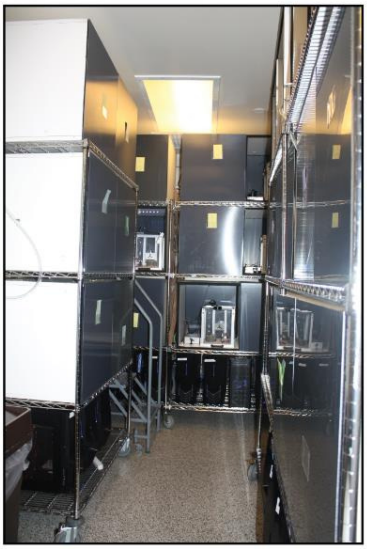


**Figure S1.4** Hardware implementation of our fully automated system. **A.** (Left) The behavior box is connected to the data acquisition card of the client computer via a general purpose breakout-box (see inset). (Right) The Box is housed in a custom-made isolation enclosure in the animal facility. **B.** Deploymenty of the system in our animal facility.

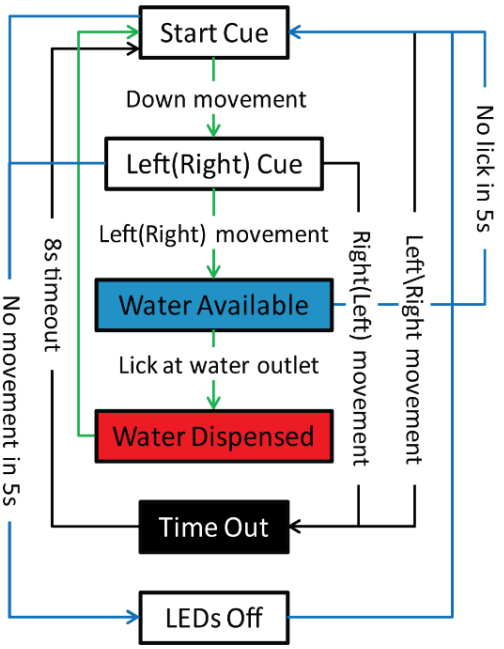



**Figure S1.5** Finite state machine for the center-out task of Figure 1.2. Each rectangle is a state and arrows indicate tranistion between states. The arrow labels specify the event causing the tranistion. Green paths correspond to a correct trials, red to incorrect and blue to abandoned ones.

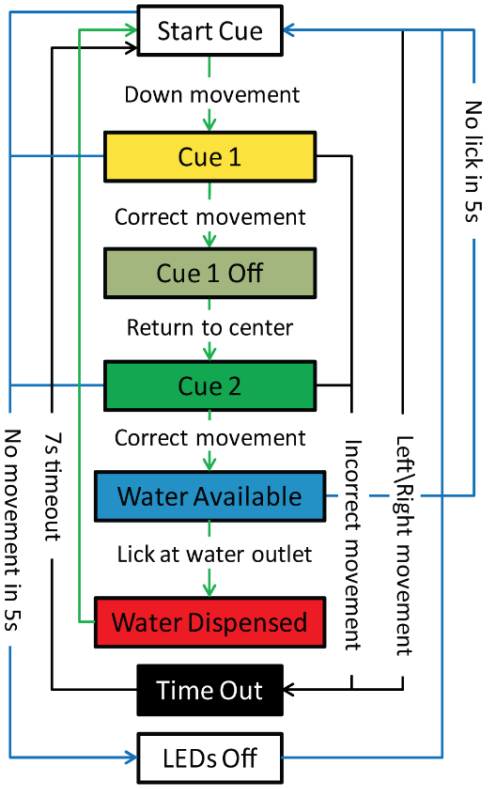



**Figure S1.6** Finite state machine for the movement sequence task of Figure 1.4. The color of the state rectangles matches the color scheme used in Figure 1.4.

## References


1. Azevedo FAC, Carvalho LRB, Grinberg LT, Farfel JM, Ferretti REL, Leite REP, et al. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. J Comp Neurol. 2009;513: 532–541. doi:10.1002/cne.21974

2. Pinto N, Cox DD, DiCarlo JJ. Why is real-world visual object recognition hard? PLoS Comput Biol. 2008;4: e27. doi:10.1371/journal.pcbi.0040027

3. Franklin DW, Wolpert DM. Computational mechanisms of sensorimotor control. Neuron. 2011;72: 425–442. doi:10.1016/j.neuron.2011.10.006

4. Sengupta B, Stemmler MB. Power Consumption During Neuronal Computation. Proc IEEE. 2014;102: 738–750. doi:10.1109/JPROC.2014.2307755

5. Teh S-Y, Lin R, Hung L-H, Lee AP. Droplet microfluidics. Lab Chip. 2008;8: 198–220. doi:10.1039/b715524g

6. Thorsen T, Maerkl SJ, Quake SR. Microfluidic large-scale integration. Science. 2002;298: 580–584. doi:10.1126/science.1076996

7. Fan HC, Wang J, Potanina A, Quake SR. Whole-genome molecular haplotyping of single cells. Nat Biotechnol. 2011;29: 51–57. doi:10.1038/nbt.1739

8. Kodandaramaiah SB, Boyden ES, Forest CR. In vivo robotics: the automation of neuroscience and other intact-system biological fields. Ann N Y Acad Sci. 2013;1305: 63–71. doi:10.1111/nyas.12171

9. Crane MM, Stirman JN, Ou C-Y, Kurshan PT, Rehg JM, Shen K, et al. Autonomous screening of C. elegans identifies genes implicated in synaptogenesis. Nat Methods. 2012;9: 977–980. doi:10.1038/nmeth.2141

10. Furlong EE, Profitt D, Scott MP. Automated sorting of live transgenic embryos. Nat Biotechnol. 2001;19: 153–156. doi:10.1038/84422

11. Pardo-Martin C, Chang T-Y, Koo BK, Gilleland CL, Wasserman SC, Yanik MF. High-throughput in vivo vertebrate screening. Nat Methods. 2010;7: 634–636. doi:10.1038/nmeth.1481

12. Denk W, Horstmann H. Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. PLoS Biol. 2004;2: e329. doi:10.1371/journal.pbio.0020329

13. Platt ML, Glimcher PW. Neural correlates of decision variables in parietal cortex. Nature. 1999;400: 233–238. doi:10.1038/22268

14. Tanji J. Sequantial Organization of Multiple Movements: Involvement of Cortical Motor Areas. Annu Rev Neurosci. 2001;24: 631–651.

15. Shadlen MN, Newsome WT. Neural Basis of a Perceptual Decision in the Parietal Cortex (Area LIP) of the Rhesus Monkey. J Neurophysiol. 2001;86: 1916–1936.

16. Newsome WT, Britten KH, Movshon JA. Neuronal correlates of a perceptual decision. 1989;341: 52–54. doi:10.1038/341052a0

17. Goodman S, Check E. Animal experiments: The great primate debate. Nature. 2002;417: 684–687. doi:10.1038/417684a

18. Zeeb FD, Robbins TW, Winstanley CA. Serotonergic and Dopaminergic Modulation of Gambling Behavior as Assessed Using a Novel Rat Gambling Task. Neuropsychopharmacology. 2009;34: 2329–2343. doi:10.1038/npp.2009.62

19. Murphy RA, Mondragón E, Murphy VA. Rule Learning by Rats. Science. 2008;319: 1849–1851. doi:10.1126/science.1151564

20. Viana DS, Gordo I, Sucena É, Moita MAP. Cognitive and Motivational Requirements for the Emergence of Cooperation in a Rat Social Game. PLoS ONE. 2010;5: e8483. doi:10.1371/journal.pone.0008483

21. Blaisdell AP, Sawa K, Leising KJ, Waldmann MR. Causal Reasoning in Rats. Science. 2006;311: 1020–1022. doi:10.1126/science.1121872

22. Zoccolan D, Oertelt N, DiCarlo JJ, Cox DD. A rodent model for the study of invariant visual object recognition. Proc Natl Acad Sci. 2009;106: 8748–8753. doi:10.1073/pnas.0811583106

23. Uchida N, Mainen ZF. Speed and accuracy of olfactory discrimination in the rat. Nat Neurosci. 2003;6: 1224–1229. doi:10.1038/nn1142

24. Kepecs A, Uchida N, Zariwala HA, Mainen ZF. Neural correlates, computation and behavioural impact of decision confidence. Nature. 2008;455: 227–231. doi:10.1038/nature07200

25. Raposo D, Sheppard JP, Schrater PR, Churchland AK. Multisensory Decision-Making in Rats and Humans. J Neurosci. 2012;32: 3726–3735. doi:10.1523/JNEUROSCI.4998-11.2012

26. Olveczky BP. Motoring ahead with rodents. Curr Opin Neurobiol. 2011;21: 571–578. doi:10.1016/j.conb.2011.05.002

27. Slutzky MW, Jordan LR, Bauman MJ, Miller LE. A new rodent behavioral paradigm for studying forelimb movement. J Neurosci Methods. 2010;192: 228–232. doi:10.1016/j.jneumeth.2010.07.040

28. Brown VJ, Bowman EM. Rodent models of prefrontal cortical function. Trends Neurosci. 2002;25: 340–343. doi:10.1016/S0166-2236(02)02164-1

29. Nudo RJ, Frost S. The Evolution of Motor Cortex and Motor Systems. Evolutionary Neuroscience; Ed Jon H Kaas. 1st ed. Academic Press; 2009.

30. Sereno MI, Allman JM. Cortical Visual Areas in Mammals (ed. A.G. Leventhal). The Neural Basis of Visual Function. Macmillan, London; 1991. pp. 160–172.

31. Erlich JC, Bialek M, Brody CD. A Cortical Substrate for Memory-Guided Orienting in the Rat. Neuron. 2011;72: 330–343. doi:10.1016/j.neuron.2011.07.010

32. Dombeck DA, Khabbaz AN, Collman F, Adelman TL, Tank DW. Imaging Large-Scale Neural Activity with Cellular Resolution in Awake, Mobile Mice. Neuron. 2007;56: 43–57. doi:10.1016/j.neuron.2007.08.003

33. Ghosh KK, Burns LD, Cocker ED, Nimmerjahn A, Ziv Y, Gamal AE, et al. Miniaturized integration of a fluorescence microscope. Nat Methods. 2011;8: 871–878. doi:10.1038/nmeth.1694

34. Zhang F, Aravanis A, Adamantidis A, de Lecea L, Deisseroth K. Circuit-breakers: optical technologies for probing neural signals and systems. Nat Rev Neurosci. 2007;8: 577–581. doi:10.1038/nrn2192

35. Luo L, Callaway E, Svoboda K. Genetic dissection of neural circuits. Neuron. 2008;57: 634–660. doi:10.1016/j.neuron.2008.01.002

36. Crabbe JC, Wahlsten D, Dudek BC. Genetics of Mouse Behavior: Interactions with Laboratory Environment. Science. 1999;284: 1670–1672. doi:10.1126/science.284.5420.1670

37. Hurst JL, West RS. Taming anxiety in laboratory mice. Nat Methods. 2010;7: 825–826. doi:10.1038/nmeth.1500

38. Clark RE, Reinagel P, Broadbent NJ, Flister ED, Squire LR. Intact Performance on Feature-Ambiguous Discriminations in Rats with Lesions of the Perirhinal Cortex. Neuron. 2011;70: 132–140. doi:10.1016/j.neuron.2011.03.007

39. Mechan AO, Wyss A, Rieger H, Mohajeri MH. A comparison of learning and memory characteristics of young and middle-aged wild-type mice in the IntelliCage. J Neurosci Methods. 2009;180: 43–51. doi:10.1016/j.jneumeth.2009.02.018

40. Gess A, Schneider DM, Vyas A, Woolley SMN. Automated auditory recognition training and testing. Anim Behav. 2011;82: 285–293. doi:10.1016/j.anbehav.2011.05.003

41. Bari A, Dalley JW, Robbins TW. The application of the 5-choice serial reaction time task for the assessment of visual attentional processes and impulse control in rats. Nat Protoc. 2008;3: 759–767. doi:10.1038/nprot.2008.41

42. Schaefer AT, Claridge-Chang A. The surveillance state of behavioral automation. Curr Opin Neurobiol. 2012;22: 170–176. doi:10.1016/j.conb.2011.11.004

43. Harel D. Statecharts: a visual formalism for complex systems. Sci Comput Program. 1987;8: 231–274. doi:10.1016/0167-6423(87)90035-9

44. Georgopoulos AP, Schwartz AB, Kettner RE. Neuronal population coding of movement direction. Science. 1986;233: 1416–1419.

45. Moran DW, Schwartz AB. Motor Cortical Representation of Speed and Direction During Reaching. J Neurophysiol. 1999;82: 2676–2692.

46. Tanji J, Shima K. Role for supplementary motor area cells in planning several movements ahead. Nature. 1994;371: 413–416. doi:10.1038/371413a0

47. Chida Y, Sudo N, Mori J, Kubo C. Social isolation stress impairs passive avoidance learning in senescence-accelerated mouse (SAM). Brain Res. 2006;1067: 201–208. doi:10.1016/j.brainres.2005.10.042

48. Ouchi H, Ono K, Murakami Y, Matsumoto K. Social isolation induces deficit of latent learning performance in mice: a putative animal model of attention deficit/hyperactivity disorder. Behav Brain Res. 2013;238: 146–153. doi:10.1016/j.bbr.2012.10.029

49. Crawley JN. Behavioral Phenotyping Strategies for Mutant Mice. Neuron. 2008;57: 809–818. doi:10.1016/j.neuron.2008.03.001

50. Poddar R, Kawai R, Ölveczky BP. A fully automated high-throughput training system for rodents. PloS One. 2013;8: e83171. doi:10.1371/journal.pone.0083171

51. Lütcke H, Margolis DJ, Helmchen F. Steady or changing? Long-term monitoring of neuronal population activity. Trends Neurosci. 2013;36: 375–384. doi:10.1016/j.tins.2013.03.008

52. Todorov E. Direct cortical control of muscle activation in voluntary arm movements: a model. Nat Neurosci. 2000;3: 391–398. doi:10.1038/73964

53. Rousche PJ, Normann RA. Chronic recording capability of the Utah Intracortical Electrode Array in cat sensory cortex. J Neurosci Methods. 1998;82: 1–15.

54. Tolias AS, Ecker AS, Siapas AG, Hoenselaar A, Keliris GA, Logothetis NK. Recording chronically from the same neurons in awake, behaving primates. J Neurophysiol. 2007;98: 3780–3790. doi:10.1152/jn.00260.2007

55. Greenberg PA, Wilson FAW. Functional stability of dorsolateral prefrontal neurons. J Neurophysiol. 2004;92: 1042–1055. doi:10.1152/jn.00062.2004

56. Schmitzer-Torbert N, Redish AD. Neuronal activity in the rodent dorsal striatum in sequential navigation: separation of spatial and reward responses on the multiple T task. J Neurophysiol. 2004;91: 2259–2272. doi:10.1152/jn.00687.2003

57. Nicolelis MAL, Dimitrov D, Carmena JM, Crist R, Lehew G, Kralik JD, et al. Chronic, multisite, multielectrode recordings in macaque monkeys. Proc Natl Acad Sci U S A. 2003;100: 11041–11046. doi:10.1073/pnas.1934665100

58. Komiyama T, Sato TR, O'Connor DH, Zhang Y-X, Huber D, Hooks BM, et al. Learning-related fine-scale specificity imaged in motor cortex circuits of behaving mice. Nature. 2010;464: 1182–1186. doi:10.1038/nature08897

59. Huber D, Gutnisky DA, Peron S, O'Connor DH, Wiegert JS, Tian L, et al. Multiple dynamic representations in the motor cortex during sensorimotor learning. Nature. 2012;484: 473–478. doi:10.1038/nature11039

60. Tian L, Akerboom J, Schreiter ER, Looger LL. Neural activity imaging with genetically encoded calcium indicators. Prog Brain Res. 2012;196: 79–94. doi:10.1016/B978-0-444-59426-6.00005-7

61. Santhanam G, Linderman MD, Gilja V, Afshar A, Ryu SI, Meng TH, et al. HermesB: a continuous neural recording system for freely behaving primates. IEEE Trans Biomed Eng. 2007;54: 2037–2050. doi:10.1109/TBME.2007.895753

62. Szuts TA, Fadeyev V, Kachiguine S, Sher A, Grivich MV, Agrochão M, et al. A wireless multi-channel neural amplifier for freely moving animals. Nat Neurosci. 2011;14: 263–269. doi:10.1038/nn.2730

63. Borton DA, Yin M, Aceros J, Nurmikko A. An implantable wireless neural interface for recording cortical circuit dynamics in moving primates. J Neural Eng. 2013;10: 026010. doi:10.1088/1741-2560/10/2/026010

64. Grand L, Ftomov S, Timofeev I. Long-term synchronized electrophysiological and behavioral wireless monitoring of freely moving animals. J Neurosci Methods. 2013;212: 237–241. doi:10.1016/j.jneumeth.2012.10.008

65. Hasegawa T, Fujimoto H, Tashiro K, Nonomura M, Tsuchiya A, Watanabe D. A wireless neural recording system with a precision motorized microdrive for freely behaving animals. Sci Rep. 2015;5: 7853. doi:10.1038/srep07853

66. Harrison RR, Kier RJ, Chestek CA, Gilja V, Nuyujukian P, Ryu S, et al. Wireless neural recording with single low-power integrated circuit. Neural Syst Rehabil Eng IEEE Trans On. 2009;17: 322–329.

67. Eliades SJ, Wang X. Chronic multi-electrode neural recording in free-roaming monkeys. J Neurosci Methods. 2008;172: 201–214. doi:10.1016/j.jneumeth.2008.04.029

68. Venkatraman S, Jin X, Costa RM, Carmena JM. Investigating Neural Correlates of Behavior in Freely Behaving Rodents Using Inertial Sensors. J Neurophysiol. 2010;104: 569–575. doi:10.1152/jn.00121.2010

69. Ferguson JE, Boldt C, Redish AD. Creating low-impedance tetrodes by electroplating with additives. Sens Actuators Phys. 2009;156: 388–393. doi:10.1016/j.sna.2009.10.001

70. Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. Commun ACM. 2008;51: 107–113. doi:10.1145/1327452.1327492

71. Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop Distributed File System. IEEE 26th Symp Mass Storage Syst Technol. 2010; 1–10.

72. Freeman J, Vladimirov N, Kawashima T, Mu Y, Sofroniew NJ, Bennett DV, et al. Mapping brain activity at scale with cluster computing. Nat Methods. 2014;11: 941–950. doi:10.1038/nmeth.3041

73. Calabrese A, Paninski L. Kalman filter mixture model for spike sorting of non-stationary data. J Neurosci Methods. 2011;196: 159–169. doi:10.1016/j.jneumeth.2010.12.002

74. Harris KD, Henze DA, Csicsvari J, Hirase H, Buzsáki G. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. J Neurophysiol. 2000;84: 401–414.

75. Lewicki MS. A review of methods for spike sorting: the detection and classification of neural action potentials. Netw Comput Neural Syst. 1998;9: R53–R78.

76. Quiroga RQ, Nadasdy Z, Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. Neural Comput. 2004;16: 1661–1687. doi:10.1162/089976604774201631

77. Shoham S, Fellows MR, Normann RA. Robust, automatic spike sorting using mixtures of multivariate t-distributions. J Neurosci Methods. 2003;127: 111–122.

78. Wehr M, Pezaris JS, Sahani M. Simultaneous paired intracellular and tetrode recordings for evaluating the performance of spike sorting algorithms. Neurocomputing. 1999;26–27: 1061 – 1068. doi:http://dx.doi.org/10.1016/S0925-2312(99)00105-8

79. Pouzat C, Detorakis G. SPySort: Neuronal Spike Sorting with Python. CoRR. 2014;abs/1412.6383. Available: http://arxiv.org/abs/1412.6383

80. Pouzat C, Mazor O, Laurent G. Using noise signature to optimize spike-sorting and to assess neuronal classification quality. J Neurosci Methods. 2002;122: 43–57.

81. Gold C, Henze DA, Koch C, Buzsáki G. On the origin of the extracellular action potential waveform: A modeling study. J Neurophysiol. 2006;95: 3113–3128. doi:10.1152/jn.00979.2005

82. Blatt null, Wiseman null, Domany null. Superparamagnetic clustering of data. Phys Rev Lett. 1996;76: 3251–3254.

83. Vazquez-Reina A, Huang D, Gelbart M, Lichtman J, Miller E, Pfister H. Segmentation Fusion for Connectomics. Barcelona, Spain: IEEE; 2011.

84. Kaynig V, Vazquez-Reina A, Knowles-Barley S, Roberts M, Jones TR, Kasthuri N, et al. Large-scale automatic reconstruction of neuronal processes from electron microscopy images. Med Image Anal. 2015;22: 77–88. doi:10.1016/j.media.2015.02.001

85. Slonim N, Atwal GS, Tkačik G, Bialek W. Information-based clustering. Proc Natl Acad Sci U S A. 2005;102: 18297–18302.

86. Quiroga RQ, Reddy L, Kreiman G, Koch C, Fried I. Invariant visual representation by single neurons in the human brain. Nature. 2005;435: 1102–1107. doi:10.1038/nature03687

87. Hahnloser RHR, Kozhevnikov AA, Fee MS. An ultra-sparse code underlies the generation of neural sequences in a songbird. Nature. 2002;419: 65–70. doi:10.1038/nature00974

88. Graziano M. The organization of behavioral repertoire in motor cortex. Annu Rev Neurosci. 2006;29: 105–134. doi:10.1146/annurev.neuro.29.051605.112924

89. Thach WT. Correlation of neural discharge with pattern and force of muscular activity, joint position, and direction of intended next movement in motor cortex and cerebellum. J Neurophysiol. 1978;41: 654–676.

90. Hikosaka O, Rand MK, Miyachi S, Miyashita K. Learning of sequential movements in the monkey: process of learning and retention of memory. J Neurophysiol. 1995;74: 1652–1661.

91. Nakamura K, Sakai K, Hikosaka O. Neuronal activity in medial frontal cortex during learning of sequential procedures. J Neurophysiol. 1998;80: 2671–2687.

92. Lu X, Ashe J. Anticipatory activity in primary motor cortex codes memorized movement sequences. Neuron. 2005;45: 967–973. doi:10.1016/j.neuron.2005.01.036

93. Shima K, Tanji J. Both supplementary and presupplementary motor areas are crucial for the temporal organization of multiple movements. J Neurophysiol. 1998;80: 3247–3260.

94. Shima K, Tanji J. Binary-coded monitoring of a behavioral sequence by cells in the pre-supplementary motor area. J Neurosci Off J Soc Neurosci. 2006;26: 2579–2582. doi:10.1523/JNEUROSCI.4161-05.2006

95. Scharff C, Nottebohm F. A comparative study of the behavioral deficits following lesions of various parts of the zebra finch song system: implications for vocal learning. J Neurosci Off J Soc Neurosci. 1991;11: 2896–2913.

96. Long MA, Fee MS. Using temperature to analyse temporal dynamics in the songbird motor pathway. Nature. 2008;456: 189–194. doi:10.1038/nature07448

97. Olveczky BP, Andalman AS, Fee MS. Vocal experimentation in the juvenile songbird requires a basal ganglia circuit. PLoS Biol. 2005;3: e153. doi:10.1371/journal.pbio.0030153

98. Fehér O, Wang H, Saar S, Mitra PP, Tchernichovski O. De novo establishment of wild-type song culture in the zebra finch. Nature. 2009;459: 564–568. doi:10.1038/nature07994

99. Kawai R. The role of motor cortex in the acquisition and production of learned motor sequences [Internet]. 2014. Available: http://search.proquest.com.ezp-prod1.hul.harvard.edu/docview/1557752642?accountid=11311

100. Laubach M, Wessberg J, Nicolelis MA. Cortical ensemble activity increasingly predicts behaviour outcomes during learning of a motor task. Nature. 2000;405: 567–571. doi:10.1038/35014604

101. Narayanan NS, Laubach M. Top-down control of motor cortex ensembles by dorsomedial prefrontal cortex. Neuron. 2006;52: 921–931. doi:10.1016/j.neuron.2006.10.021

102. Isomura Y, Harukuni R, Takekawa T, Aizawa H, Fukai T. Microcircuitry coordination of cortical motor information in self-initiation of voluntary movements. Nat Neurosci. 2009;12: 1586–1593. doi:10.1038/nn.2431

103. Cohen D, Nicolelis MAL. Reduction of single-neuron firing uncertainty by cortical ensembles during motor skill learning. J Neurosci Off J Soc Neurosci. 2004;24: 3574–3582. doi:10.1523/JNEUROSCI.5361-03.2004

104. Azim E, Jiang J, Alstermark B, Jessell TM. Skilled reaching relies on a V2a propriospinal internal copy circuit. Nature. 2014;508: 357–363. doi:10.1038/nature13021

105. Esposito MS, Capelli P, Arber S. Brainstem nucleus MdV mediates skilled forelimb motor tasks. Nature. 2014;508: 351–356. doi:10.1038/nature13023

106. Murray PD, Keller A. Somatosensory response properties of excitatory and inhibitory neurons in rat motor cortex. J Neurophysiol. 2011;106: 1355–1362. doi:10.1152/jn.01089.2010

107. Georgopoulos AP, Kalaska JF, Caminiti R, Massey JT. On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. J Neurosci Off J Soc Neurosci. 1982;2: 1527–1537.

108. Chestek CA, Batista AP, Santhanam G, Yu BM, Afshar A, Cunningham JP, et al. Single-neuron stability during repeated reaching in macaque premotor cortex. J Neurosci Off J Soc Neurosci. 2007;27: 10742–10750. doi:10.1523/JNEUROSCI.0959-07.2007

109. Taylor DM, Tillery SIH, Schwartz AB. Direct cortical control of 3D neuroprosthetic devices. Science. 2002;296: 1829–1832. doi:10.1126/science.1070291

110. Carmena JM, Lebedev MA, Henriquez CS, Nicolelis MAL. Stable ensemble performance with single-neuron variability during reaching movements in primates. J Neurosci Off J Soc Neurosci. 2005;25: 10712–10716. doi:10.1523/JNEUROSCI.2772-05.2005